3-2002

Categorizing Network Attacks Using Pattern Classification Algorithms

George E. Noel III

Categorizing Network Attacks Using Pattern Classification Algorithms

THESIS

George E. Noel III, First Lieutenant, USAF

AFIT/GIR/ENG/02M-03

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense, the United States Air Force, or the United States Government.

AFIT/GIR/ENG/02M-03

CATEGORIZING NETWORK ATTACKS USING PATTERN CLASSIFICATION ALGORITHMS

THESIS

Presented to the Faculty of the
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
In Partial Fulfillment of the Requirements for the
Degree of Master of Science

George E. Noel III, B.S.
First Lieutenant, USAF

March, 2002

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT/GIR/ENG/02M-03

CATEGORIZING NETWORK ATTACKS USING PATTERN CLASSIFICATION ALGORITHMS

George E. Noel III, B.S.

First Lieutenant, USAF

Approved:

Gregg H. Gunsch — 11 MAR 2002
Doctor Gregg H. Gunsch — Date
Thesis Advisor

David P. Biros — 11 MAR 2002
Lt Col David P. Biros — Date
Committee Member

12 Mar 2002
Doctor Steven C. Gustafson — Date
Committee Member

Acknowledgements

Without the dedicated guidance and support of many, this thesis would not have been possible. First, I would like to thank Dr. Gregg Gunsch, whose tireless efforts and unerring guidance were invaluable towards both this effort and my own personal development. With his expert guidance, the Information Warfare Research group will no doubt touch the field in ways beneficial to the world.

To Dr. Steven Gustafson, thank you for investing countless hours opening my eyes to the marvelous world of pattern recognition. For the time and effort you donated bringing me from a novice in the field to where I am now, you are truly an instructor at the top of your peers. I could not have done this work without your help.

And last, but certainly not least, to my family who saw only my back at the computer for eighteen months, thank you for your understanding and support. This would not have been possible without you.

George E. Noel III

Table of Contents

List of Figures

Figure Page

List of Tables

Table Page

AFIT/GIR/ENG/02M-03

Abstract

Information systems are often inundated with thousands of attack alerts and are unable to distinguish novice hacker probes from genuine threats. Pattern classification can help filter relatively benign attacks from alerts generated by anomaly detectors, limiting the number of alerts requiring attention.

This research investigates the feasibility of using pattern classification algorithms on network packet header information to classify network attacks. Both linear discriminant and radial basis function algorithms are trained using flood and scan attacks. The classifiers are then tested with unknown floods and scans to determine how well they categorize previously unseen attacks. The results indicate the radial basis function algorithm classifies most packets correctly. The linear discriminant algorithm correctly classifies half of unknown packets.

CATEGORIZING NETWORK ATTACKS USING PATTERN CLASSIFICATION ALGORITHMS

I. Background and Problem Statement

The growing dependence of the U.S. Department of Defense on information technology provides an appealing Achilles heel for hostile nation-states to strike under a cloak of anonymity. By routing connections through multiple points and hiding network information from the target, any nation could strike down or severely disrupt the information infrastructure that is becoming vital to both national defense and economic viability. Security specialist Stephen Hildreth, in a report to the Congressional Research Service, states that over 20 countries are actively developing information operations targeted at the United States [Hildreth01]. In addition to national threats, Lieutenant General Minihan, former Director of the National Security Agency, claims that even organized crime and terrorist organizations pose serious risks to national security if a coordinated attack were launched [Minihan98]. Protecting the communications assets that are, in Lieutenant General Minihan's words, "vital to all aspects of DoD operations" has high priority in the military and other government agencies. The act of protecting these systems, however, is not as easy as installing the latest off-the-shelf software. While many software tools exist for network defense, many are plagued with problems. One layer of this defense, the Intrusion Detection System (IDS), is designed to detect potential intruders trying to break into a system or those who have already gained unauthorized access. However, it is frequently easy to sneak past the detectors with new previously unseen attacks or to overwhelm them with attack information, which makes finding the real attack difficult. Machines must be utilized to first organize and simplify the information a systems administrator must handle and second, to detect previously unseen attacks.

1.1 Research Focus

This research focuses on developing pattern-classification systems to categorize attacks into predefined categories such as Flooding Denial of Service, Distributed Denial of Service, Nukers, and Portscans. Such categorization decreases the data received by systems administrators by collecting what may be hundreds of packets into one attack, decreasing the amount of information they must analyze. This categorization might also be used to detect attacks and find packets of an attack that anomaly-based IDS would recognize because of its similarity to normal traffic.

1.1.1 Problem Statement. As Internet usage grows, the amount of traffic that network security personnel must deal with can overwhelm their capabilities. Current IDSs can help alleviate some of this burden, but even they are inundated with problems. Most IDSs are based on pattern-matching, containing signatures for very specific attacks. In many cases, a slight variation on a particular attack requires human intervention to generate a new signature. With the high frequency of new attacks generated by the hacker community, ensuring signatures that recognize new attacks can be a difficult, if not impossible task.

In addition to detecting new attacks, IDSs must minimize false alarms. Systems administrators are often overworked and it may take very few additional network attacks to overwhelm them. If hundreds of attacks were launched, it may become difficult to discover which attacks were fairly benign and which could cause significant damage. Some tools have been developed with the sole purpose of overwhelming systems administrators, as will be discussed in more detail in Chapter II.

1.1.2 Objectives and Approach. The proposed system in this research has several objectives:

- **Objective 1:** Develop pattern classification algorithms that indicate the category of an attack for unknown packets.

- **Objective 2:** Determine automatically which features are most important for discriminating between attacks.
- **Objective 3:** Require little to no human intervention after the training process to properly classify attacks.

There are several ways Objective 1 can be accomplished, some involving more human intervention than others. The structural approach to pattern classification requires that the structural properties of the subject be apparent. While network traffic does have structure, one cannot assume a predefined structure that can be used to distinguish between attack and non-attack network traffic. The decision-theoretic approach involves partitioning the feature space into two or more categories with the goal of classifying future packets based on past data [Bow84]. With this approach, a proper feature set must be chosen. This research makes the assumption that there exists no best-fit feature set for all attacks. To minimize user intervention when the system is learning attack categories, the system must automatically identify which features are important for proper classification and distinction from normal traffic. Given an automatically generated set of features, the system should be able to produce a classifier for detecting new attacks that fits attacks into the predefined categories.

1.1.2.1 Algorithm Selection. When selecting an appropriate pattern classification algorithm, simplicity is important. A linear classifier can provide information on what fields it is using to classify. A three-layer neural network trained with backpropagation can classify complex patterns where a linear classifier fails, but identifying the mechanics behind the classification is difficult. This research starts with the simple linear discriminant algorithm and finishes with the more complex radial basis function algorithm. These two algorithms were chosen based on their ability to generalize and their transparency when determining how unknown data is classified.

1.1.2.2 Training and Test Data Selection. To simulate a real-world network as closely as possible, attacks and normal data are generated to closely resemble what would occur on a network. This research assembles an isolated network with two subnets, one the target network and the other a hacker's network. Attacks are launched against the target network and recorded using the TCPDump binary format. This data is formatted as necessary and used to train the classifiers. The attacks are chosen to represent a wide variety of attacks in a particular category.

Network attacks consist of a single packet or multiple packets. Some attacks use identical packet field values for each packet and intend to flood or overwhelm the victim with traffic volume. Since these fields seldom change, few packets are necessary to train the classifier. Other packets change several fields with each new packet, requiring more packets to train the classifier. Thus understanding the attack is important for properly training a classifier.

Improper training is a risk when performing pattern classification. If an attack is launched from a different subnet, the classifier may train on the IP address of the attacker. Another attacker coming from a different subnet could be improperly classified based solely on their IP address. To avoid this problem, certain fields must be eliminated to prevent improper training that would result in incorrect classification.

Once the classifier has been trained, the leave-one-out test can be performed. This test trains on n-1 attacks and uses the remaining attack to test the success or failure of the classifier. The classifier can then be trained by leaving out another attack and training on the remaining attacks. This is repeated until the algorithm has been tested on all attacks. [Duda01]

1.1.2.3 Stopping Criterion. Once the classifier has been thoroughly tested using the leave-one-out test, success or failure is indicated by the percentage of attacks successfully classified. If this percentage is low, a new classifier algorithm can be chosen. To limit scope, this research analyzes two classifiers.

1.1.3 Scope. Pattern classification is an old field compared to computer networks, and has developed many hundreds of techniques, [Duda01]. Testing even a small proportion of the available pattern classification algorithms with the network attack classification system is not feasible. Based on a literature review in Chapter II of existing algorithms and an analysis of the data used, the system is tested using linear discriminant and radial basis function pattern classification algorithms. The advantages of these two algorithms, discussed with greater detail in Chapter II, include their ability to generalize, their speed, and simplicity.

The network protocol map is an extremely heterogeneous entity. Over the years, various protocols have developed by different vendors, ranging from the widely used TCP model to lesser known protocols. While a commercial tool would need to support many protocols, the majority of network traffic and attacks can be covered using TCP, UDP, and ICMP packets. However, monitoring all fields for each protocol would greatly increase the complexity of the classifier. For this reason, and because of the large number of attacks available, TCP traffic is used to test the algorithms. Creating a system that could classify UDP and ICMP, or higher layer protocols such as SMTP or HTTP should be a matter of performing the same classification using different features.

Other header layers may exist, including ethernet headers or token ring headers depending on the link layer. Higher layer headers, such as an SMTP header for an e-mail message, may provide useful information to a pattern classification algorithm. During the research, however, a large number of features produced poor results. Since most attacks use TCP packet headers and, therefore, a large sample of attacks could be used to test the classifier, only TCP and IP packet headers were used as features.

The range of possible network attacks is as varied as network protocol types. Some attacks, such as buffer overflow attacks, look like normal traffic up to the application layer. Since the attack classification system looks only at TCP and IP headers, it will not detect these attacks. A pattern classification algorithm could be designed to handle the application layer, but would be necessarily complex given the enormous number of packet types, and, therefore, is beyond the scope of this research. Other attacks may reside in SMTP header information and therefore would not be detected by the attack classifier. An effort is made to ensure that a wide variety of attacks are selected, but attacks are also chosen that look like they may contain a pattern residing on the network layer or below, such as those that take advantage of TCP flags.

1.2 Document Overview

This document reports research performed using pattern classification for network attacks. Chapter II provides an introduction to IDSs, network attacks, pattern classification, and a synopsis of past research in pattern classification pertaining to IDSs. Chapter III describes the research methods and implementation. Chapter IV outlines the results of the experimentation and Chapter V concludes with a discussion of the research effort and recommends future directions.

II. Literature Review

This chapter provides the background necessary for understanding pattern recognition as applied to intrusion detection. Section 2.1 discusses the existing threat to information systems. Section 2.2 outlines the potential for a coordinated attack that masks dangerous attacks using relatively benign attacks which overwhelm security personnel. Section 2.3 provides a brief introduction to Intrusion Detection Systems (IDSs), followed in section 2.4 by exposure to the machine learning utilized in IDSs. Section 2.5 describes pattern classification, including the different types of pattern recognition algorithms. Next, Section 2.6 gives a brief introduction to network data, Section 2.7 concludes with an outline of existing network attacks and network categories.

2.1 Information Threat

As information systems proliferate across the world, threats grow increasingly complex. Vendors release patches with growing frequency to counter vulnerabilities discovered in their systems, yet attackers continue to find newer ways into the system. Some of these new attacks are growing in complexity and circumventing current defense measures. For instance, many firewalls are heavily protected against traffic flowing into a network but give those on the inside free access to the outside world. Once a trojan horse (a software tool that resides on a system that provides back-door access) has been installed on an internal machine, it can connect out to an external site. Since the connection is established from an internal network address to an external address, the firewall protection is circumvented. The attacker can now use this stream to command trojan horses without the need to connect into the site, since the trojan horses connects out to them. Several of these trojan horses are available for download, including Trinity, Strackeldralt, and Entitee (http://www.nipc.gov/warnings/advisories/2000/00-055.htm). In addition, standard viruses that used to be transmitted by floppy disk are now automatically propagating themselves over the internet. The Nimda virus can infect other machines using e-mail, web vulnerabilities, open network shares, and previously established back-doors (http://www.cert.org/advisories/CA-2001-26.html).

In addition to novel complex attacks, recent attack tools often provide methods to gradually probe a target over a relatively long period of time. This virtually silent attack can often slip through IDSs designed to detect noisier attacks that take place within a short period of time. This problem, coupled with the ingenuity of new attacks, can create a challenge when developing automated tools to counter the threats.

2.2 Overwhelming Coordinated Attacks

While attacks have grown more complex, the knowledge required to launch pre-packaged attacks has decreased, which can create a problem for systems administrators who find themselves wading through attack after attack by what are commonly referred to as "script kiddies"-hackers who know just enough to launch a prefabricated script at a target. As Major General John Campbell states, "We have found that almost anyone with a computer and modem can use specialized malicious software and tools and attempt to disrupt our network operations and make it difficult for us to effectively carry out our missions," [Peters99]. While current attacks are fairly unorganized, an organized attack could flood and easily overwhelm network security personnel with a barrage of packets, effectively hiding real attacks behind a screen of decoy attacks.

Ptacek and Newsham's paper "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection", describes methods for evading or overwhelming IDSs by using up their resources. Even if a Denial of Service (DoS) attack upon an IDS did not succeed in using up resources, the massive number of reported attacks could make finding the true attack a daunting task [Ptacek98]. With often thousands of attacks reported, systems administrators would (in many cases) fix any damage and give up on trying to locate how the attacker got in to close the hole.

2.3 Automated Intrusion Detection

Most intrusion detection systems follow a form of pattern matching. Attack signatures are generated manually to match a certain pattern, and when packets are discovered that match those patterns, system administrators are notified. There are many disadvantages to current signature-based IDS. First, for each new attack a signature must be created, which often occurs days or weeks after the first instance of the attack is detected, and takes even longer to implement and disseminate to the community. Second, the systems, in most cases, have a very small window of detection that only catch attacks patterns occurring during a short period of time. While the inexperienced may blast at a target, the professional hackers will evade the IDS by probing slowly over time. Finally, the signature database can grow uncontrollably large as the number of required signatures for new attacks increase. This increases system complexity and can slow down the analysis of real-time traffic [Stillerman99]. Also, minor changes in the construction of the attack can elude all signatures while still preserving functionality.

Anomaly-based systems try to detect attacks by taking a 'snapshot' of normal traffic. Anything that falls outside the bounds of normality is considered a potential attack, which overcomes the problems signature-based systems have detecting novel attacks. Assuming the novel attacks differ from normal traffic, they will appear as anomalies.

One such anomaly detector, the Artificial Immune System, models its architecture after the human body's immune system [Forrest97]. Antibodies are developed by comparing with self (normal traffic) and rejecting those antibodies that match self. Once developed, any packets that fall within an antibody, or if using costimulation, two antibodies, are flagged as possible attacks. [Williams01]

Anomaly-detectors are not without weaknesses. They are often plagued with false positives (false alarms). Since the landscape of network traffic changes frequently with the changing needs of the community, the detector must change with it. Unfortunately, each change produces many false alarms until the system is retrained. An attacker could use this retraining as an opportunity to ensure that the attack data is accepted as normal data and could then have free access to the network.

2.4 Pattern Classification

Pattern classification involves predicting future data based on past data. It is also known (in key aspects) as pattern recognition, discriminant analysis, decision theory, assignment analysis, etc [James85].

2.4.1 Classifier Design. The success of a classifier is highly dependent upon the data and features chosen. For this reason, the design process that precedes actual classification can be considered more important than selecting the 'correct' classification algorithm. The following sections outline classifier design, including sensing, segmentation, feature extraction, classification, post-processing, and finally decision [Duda01].

2.4.1.1 Sensing. Sensing records data to be used for classification. It may involve changing light waves into binary information, as in a face recognition tool. In this research, the sensing tool is a TCPDump sniffer that listens for network traffic and records the attributes of the packets for analysis. Once in machine-readable form, the packets are ready for the next step.

2.4.1.2 Segmentation. Segmentation determines where each element begins and ends in the classification process. The segmentation process in face recognition, for example, identifies how the face is positioned – e.g., upside-down, shaded, slightly tilted, or to the left or right side of the image – and determines the boundaries of the face even when positioned in abnormal ways [Duda01]. However, for network data the packet elements must already be clearly defined, or the destination computer could not reassemble them into meaningful data. This makes the segmentation phase trivial for the IDS pattern classifier.

2.4.1.3 Feature Extraction. Feature extraction can be a complicated aspect of pattern recognition. Facial pattern recognition tools must rotate the face, change shading, adjust for facial expressions, and account for a myriad of other differences that could influence the accuracy of the classifier. When recognizing network patterns, the task is easier since the features are packet values and, therefore, already quantified. However, initial testing during algorithm development indicated that raw network data produced a poor classifier.

Some adjustments can be made to simplify the classification process and make it more effective. First, when using a linear classifier, the data can be shifted so that the origin of the multidimensional space, or the data mean, is at zero. Second, fields in a network packet are inherently dissimilar in range. The first byte of an IP source address has a range of 0-255 while the sequence number has a range of 0-4294967296. The differences in range can significantly impact the accuracy of a classifier. Subtracting the mean and dividing the result by the standard deviation can normalize these values so they cover the same range and no single field will have undo influence over the results.

Many pattern classification algorithms, to include linear discriminant and radial basis function, assume some notion of metric in the features. Obviously, 10 is closer to 100 than 900. However, if a field utilized a number to indicate color, it would be meaningless to state that the 2nd color is closer to the 4th color than the 6th color. [Duda01] For this reason, features should be extracted so they represent metric data and can be used effectively in a linear discriminant or radial basis function algorithm.

2.4.1.4 Classification. Classification is the process of selecting and using a classifier that fits the data. Different classifiers are discussed in later sections, including those that best fit the network data.

2.4.1.5 Post-Processing. Post processing analyzes and performs some action with the results of the classifier. This action may signal the user about a possible anomaly and its classification, or it may simply block unauthorized network access. In addition, post processing may provide feedback to the classifier to help detect future attacks, (as in a backpropagation-trained neural network) [Duda01].

2.4.2 Classification Algorithms. Classification algorithms vary depending on the data to be classified. Non-metric algorithms, such as decision trees or grammatical methods, are best suited to problems that lack a measure of distance between vectors, such as colors or fruit types [Duda01]. Most packet header fields are numerical and contain some level of distance. For instance, the lower network ports listen for basic services while many hacker trojan horses utilize higher ports. For this reason, this section looks primarily at metric classification algorithms.

2.4.2.1 Bayes Decision Theory. Bayes Decision theory, which is basic to pattern classification, provides rules for classifying an entity when there is prior knowledge. For instance, in network data a packet has various known attributes when detected on the network. While the network administrator may not know if the packet is contained in an attack or in normal network data, he may know that when the urgent pointer is set to 666, there is a much higher probability that the packet is an attack. Bayes rule signifies this as:

$$P(a|t) = \frac{P(t|a)P(a)}{P(t)}, \tag{2.1}$$

where $P(a|t)$ is the probability that the network is under attack a given that attribute t is observed, $P(t|a)$ is the probability of t given a, $P(a)$ is the prior probability of a, and $P(t)$ is the prior probability of t. While simple in theory, the application of this rule is usually not very straightforward. Finding the classifier that minimizes error (and hence, maximizes the predictability of future data) can be difficult [Fukunaga90]. For example, all the probabilities of certain attribute settings should be known. In a high dimensional space, obtaining this knowledge may be impractical. A given Bayes classifier may not continue to be the best classifier as new data comes in or as external factors change.

2.4.2.2 Linear Discriminant Function. Classifying real-world data usually involves settling for a less than optimal solution. One of the simpler forms of pattern recognition uses a linear decision boundary, often in a multidimensional space. In general, the goal of a linear discriminant is to create a hyperplane decision threshold through an n-dimensional space that provides the optimal separation between two classes.

As it is the workhorse of classification, there are many linear discriminant approaches. This section touches on the diversity of potential techniques by examining two linear discriminants.

Minimum Mahalanobis-distance. If normality of the data can be assumed, then the minimum-Mahalanobis-distance classifier provides a useful linear discriminant function. This classifier involves calculating the Mahalanobis distance D from the mean of each class, and if $D(Class_1) > D(Class_2)$, for a new data point, then it is classified as class two. The Mahalanobis distance is the Euclidian distance weighted by the covariance matrix of the data [Duda01].

Fisher's Linear Discriminant. The minimum-Mahalanobis-distance classifier assumes normality for the density function. However, in many cases this assumption is not valid, such as with network data, and other classifiers must be used. The Fisher linear discriminant function is often effective for high-dimensional problems where normality cannot be assumed [Bishop95]. It attempts to project all points onto a single vector, essentially reducing a multidimensional problem to one dimension. The slope of the vector can have significant impact on the accuracy of the classifier, as indicated in Figure 2.1.

The Fisher linear discriminant can be represented by $y = w^t x$ where y is the position of the multidimensional point x projected onto the w vector. When divided into two segments, this

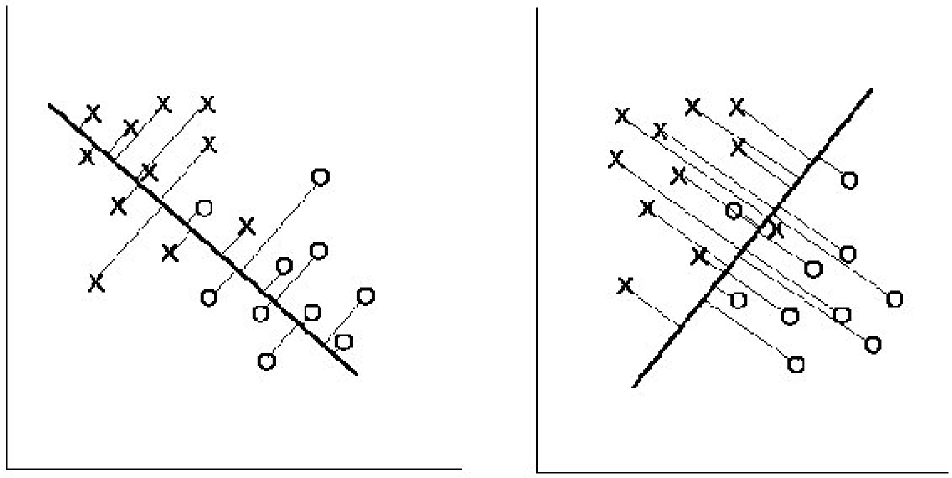


Figure 2.1 A good linear classifier (Left) and a poor linear classifier (Right). The points are mapped to the vector with perpendicular lines and a decision threshold is chosen along the vector so that most points in the two categories (X and O) are separated by the threshold.

provides optimum classification in that the data points mapped onto it tend to be in one segment for one class and the other segment for the other class. Finding w involves maximizing the ratio of the between-scatter matrix S_B and the within-scatter matrix S_W. The between-scatter matrix measures the squared distance of the means of the classes, and the within-scatter matrix measures the sum of the variances of the classes. Thus a small within-scatter matrix and a large between-scatter matrix provides the best discrimination, and maximizing J,

$$J(w) = \frac{w^t * S_B * w}{w^t * S_W * w} \tag{2.2}$$

ensures the best linear classifier [Duda01]. The S_W can be calculated by summing the scatter matrix from two classes, $S_W = S_1 + S_2$. The scatter matrices can be calculated with $S_W = (x_1 - m_1)(x_1 - m_1)T + (x_2 - m_2)(x_2 - m_2)T$ where x_1 and x_2 are vectors of the multi-dimensional points in the two classes and m_1 and m_2 are the means of the classes.

2.4.2.3 Parzen Window Distributions. The Parzen window provides a method for estimating future data point categories based on past distributions. It centers a Gaussian probability distribution function at each data point and sums the functions divided by the number of data points. By adjusting the standard deviation to higher values, the data population can be represented statistically with few data points, while adjusting it to lower values can better represent smaller sample sets without making predictions about the population. [Duda01]

2.4.2.4 Quadratic Discriminant Function. If there are large differences in covariance matrices, then a quadratic discriminant function may be indicated. While the linear discriminant tries to fit a line through a multidimensional space that best discriminates between two or more classes, the quadratic discriminant function tries to fit a curve in a multidimensional space that best discriminates between classes. Quadratic classifiers frequently have a much higher bias than linear classifiers, especially when there are few samples for training [Fukunaga90].

2.4.2.5 Neural Networks. Linear and quadratic discriminators are not suited for all pattern recognition problems. For instance, a simple linear or quadratic discriminant function cannot solve the XOR (exclusive-or) problem. In this problem, if the inputs are 1 and 1 or 0 and 0, the output is 0; however, if the inputs are 0 and 1 or 1 and 0, the output is 1. As Figure 2.2 shows, a straight line or even a curved line can not separate the 0 (white) and 1 (black). This can also be illustrated with continuous rather than discrete scales, as indicated in Figure 2.3. Drawing

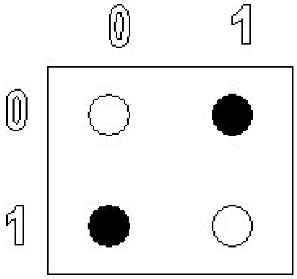


Figure 2.2 The XOR problem with a discrete scale.
A linear decision threshold cannot be drawn to perfectly discriminate the dark from light circles.

a curve that perfectly divides the dark from light regions is not possible, and this problem is of the type for which multilayer neural networks are useful. The lower layer usually consists of one or more neurons, each implementing a nonlinear function with multiple inputs and a single output. The outputs from the first layer are usually used as inputs for a second layer, which is the actual classifier function [Duda01].

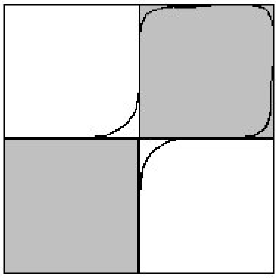


Figure 2.3 The XOR problem with a continuous scale.
A linear decision threshold cannot be drawn to perfectly discriminate the dark from light.

Using a neural network, nearly any discriminant function can be implemented. In fact, it has been shown that any function that is continuous from input to output can be classified given enough hidden units [Duda01]. However, designing a neural network to solve a particular problem is usually not a trivial matter. In addition, better fit usually comes at the expense of generalizability.

In the multilayer neural network, each hidden unit output is passed onto the next layer with a weight assigned to it. The weights are often determined by the backpropagation algorithm, which uses classification error to determine weight changes in an iterative procedure.

In IDS research, neural networks trained using backpropagation are popular when looking for patterns that might signify unwanted behavior by users. Such neural networks have been used in host-based IDSs to detect abnormal user behavior [Ghosh99, Bonifacio98]. Network-based IDS studies using neural networks have been limited, with Lee and Heinbuch's research looking for violations of network protocol being one of the few [Lee01].

One of the main drawbacks to backpropagation-trained neural networks is that it is difficult to discern why a classifier succeeds or fails to properly classify. The internal structure is often too complex to be dissected for analysis [Bishop95]. For this research, using a backpropagation-trained neural network would fail to indicate if the classifier trained solely on one field. A single field may be easy to modify, effectively hiding an attack from the classifier.

2.4.2.6 Radial Basis Function Networks. The radial basis function estimates densities using localized Gaussian probability distributions. Where the Parzen window typically uses Gaussian probability functions to estimate a distribution along a single dimension, the radial basis function measures density in a multidimensional space to best predict the classification of future data points [Duda01].

The radial basis function uses the equation $y = W * E$, where E is an n by n Gaussian function distribution matrix and n is the number of packets submitted to the classifier. To train the classifier, a separate equation is used for each category. Finding E during training involves solving

$$E_{i,j} = e^{\frac{\sum_{k=1}^{x}(M_{k,i}-M_{k,j})^2}{2\sigma^2}} \tag{2.3}$$

where $M_{i,j}$ is the matrix of j packets with i fields. The y vector contains n elements, where $y_j = 1$ if the corresponding packet in $M_{i,j}$ is of that category and a -1 if it is not. Solving this equation for W, we can get the coefficient weight vector with $W = y * E$. The resulting W vector of n elements can be used to classify future data. Inserting an unknown packet into the formula yields a vector E multiplied by the coefficient vector W to produce a scalar y value. E is calculated with

$$E_j = e^{\frac{\sum_{k=1}^{x}(Unkk-M_{k,j})^2}{2\sigma^2}} \tag{2.4}$$

where Unk is the unknown packet vector. The resulting y scalar value can be compared with the resulting y value from the other categories. The category with the highest y value is most likely the proper classification of the unknown packet. [Duda01]

The radial basis function is suitable for complex problems that the linear discriminant can't handle. However, it is limited by the computational capacity. Since taking the inverse of the E matrix can be computationally expensive, the matrix should be kept small [Duda01]. While a larger number of packets would train the classifier on a wider range of traffic that it may have to classify, it will also slow down the calculation.

2.4.3 Singularity Problems.

In most pattern classification algorithms discussed in this chapter, an inverse matrix is calculated. In certain situations, taking an inverse of a matrix can result in a singular matrix. By definition, a matrix A is invertible if we can find a unique matrix B such that $AB = BA = I$ [Herstein88]. During algorithm development, it was discovered that duplicate rows in a matrix frequently arise and yield a singular matrix. In addition, too little variation between packets can cause a singular matrix. Since it becomes impossible to continue calculations with a singular matrix, avoiding this problem is important.

2.5 Network Data

Throughout the history of computer networks, many protocols have emerged. With few exceptions, most faded into disuse and are rarely used today. The Department of Defense's TCP/IP protocol is one that has thrived since its introduction. This protocol consists of four layers, each providing encapsulation of the upper layers and data hiding of the lower layers. From highest to lowest, the layers consist of the Application layer, Transport layer, Internet layer, and finally the Physical or Network access layer. Since it accounts for the largest percentage of network traffic on the internet, it is the protocol used in this research.

2.5.1 Application Layer. The Application layer provides end-to-end communications for various applications. From File Transfer Protocol (FTP) to HyperText Transport Protocol (HTTP) or web traffic, this layer consists of data streams concerned with passing data to some destination. Any attacks that appear in this layer may vary significantly in size, and detecting them can be difficult. The easy solution is to perform pattern matching using a database of attack signatures. This has the problem of not being able to detect new attacks. Others have tried pattern recognition approaches by either recording and searching for attack patterns, hoping that trained attacks will generalize to unknown attacks[Bonifacio98], or recording patterns in user behavior and flagging when the user varies from the normal pattern [Ryan98, Ghosh99]. Searching the stream for patterns that could appear anywhere is a challenging problem with no simple solutions. Many attacks are hidden from all layers except the application layer, so while difficult, this is an important problem.

2.5.2 Transport Layer. The Application layer simply passes data to the Transport layer. Here it is broken up into packets, each with enough information to route the packet towards the destination. At this layer, the packet can be a TCP packet or a UDP packet. With TCP, lost packets are noticed and retransmitted. Lost UDP packets are not noticed, making this format suitable for those applications where a few lost packets won't significantly affect performance. The Transport layer header contains many fields that help establish connections and specify the type of service the packet belongs to, such as FTP or HTTP. Many attacks manifest themselves at this layer, using the fields in the TCP or UDP packet to disrupt service or gain unauthorized access. They may also use a large number of packets to bog down a connection or force it to crash. This layer and the following Internet layer are be the focus layers of this research.

2.5.3 Internet Layer. The Internet layer contains useful information for getting the packet to the proper end machine. This layer is responsible for end to end communications routing of packets and includes a special class of error packets, called ICMP packets. Some attacks will use these fields in the IP header to disrupt service.

2.5.4 Network Access Layer.

The Network Access layer is often further broken up into the Physical and Data Link layer. The Physical layer includes the wires and hardware that help move the data. The Data Link layer is concerned with getting each packet to the next hop. The Network Access header is usually stripped off at each hop [Stevens94]. This means an attacker must be on the same subnet as the target to utilize flaws in the Network Access layer. For this reason, few attacks exist that utilize the Network Access layer headers. The lack of attacks can make developing a sufficiently large data set difficult.

2.6 Network-Based Attacks

As the Internet has matured, the number of attacks available to hackers has grown significantly. This section outlines the various categories and possible classifications of attacks found in current literature. While literature in the area of attack classification categorizes many attacks by the resulting damage or information compromised, this may not be possible for an automatic pattern recognition algorithm. An algorithm trained on packet header information must classify using similarities in the header fields. For this reason, categories must be developed based on predicted similarities in attacks.

2.6.1 Information-Gathering Scans.

Before an attack can be launched, most hackers gather information about potential targets using many different techniques. Frequently, those indiscriminately searching for potential targets scan popular Internet provider subnets. According to literature in the intrusion analysis field, information-gathering techniques can be divided into scans for services and data enumeration [Northcutt01, Scambray01]. This categorization is based upon the methods used to gather the information and the type of information gathered by these methods.

2.6.1.1 Scans For Services.

Northcutt et al. categorizes scans for services under a broad category of network mapping. This category covers any network activity designed to provide the potential attacker with a layout of the network, including what services are running [Northcutt01]. The categorization relies heavily on the intent of the user, since a telnet connection used to map the network can also provide data enumeration information. Automated IDSs are frequently not effective at determining attacker intent; therefore, this would not be an effective categorization method.

Categorizing scans by the network method used to gather information provides a cleaner delineation between information enumeration and network scans for services. A scan would provide information on the active machines on the network and the services running on that particular machine based on the open ports. Data enumeration is limited to a single probe that provides information over the application layer, such as Operating System version or the type of service daemon running.

A network scan can come in many different forms. Any multipacket scan can be defined as either a horizontal scan or a vertical scan. A horizontal scan looks at a particular port across a wide range of computers while a vertical scan looks at many ports on a single computer [Staniford00]. Most scans use features of the packet header to elicit a response from the targets. Perhaps the toughest to identify with an anomaly detector is the TCP connect scan, since it uses standard methods for establishing a complete connection to the target host. Since it looks very similar to normal traffic, one of the only indications is the time the connection was active, coupled with the number of connections established to ports or to a particular port. Other scans, such as FIN or ACK scans, use a partial connection or malformed packet to force the host to give a response. Since these packets are not normal for network traffic, detection using an anomaly-detector is easier.

2.6.1.2 Enumeration.

Data enumeration involves gathering information about a particular host or network that may prove useful in an attack. Scambray et al. categorizes the type of information that can be gathered under three categories: network resources and shares, users and groups, and applications and banners [Scambray01]. Network resources and shares may be gathered by exploiting weaknesses in the popular NetBIOS protocol used in Microsoft OSs. In addition, NetBIOS can provide user and group information helpful in identifying user accounts to crack. Finger is an old but frequently effective tool for determining user accounts on a Unix system. Telnet is an effective tool for gathering banner information, or the information that is given on initial connect such as daemon version number, from many different connection-based services on a network.

Enumeration activity differs from scanning activity in that it provides information beyond simply indicating which ports are open and what IP addresses are in use [Scambray01]. Since most of the enumeration information exists on the application layer, tools that analyze at the transport layer or below may have trouble distinguishing the enumeration attack from normal traffic.

2.6.2 Denial of Service (DoS). A denial of service attack degrades or terminates network services in some manner. Northcutt et al. places denial of service attacks in two categories based on their end result: resource starvation and bandwidth consumption. Scambray et al. defines two other categories: Programming flaws or nukers, and routing or DNS attacks.

2.6.2.1 Resource Starvation. Resource starvation usually involves using up available ports, memory, or filling up a hard drive. In many TCP services, a four-way handshake is required to complete a connection. The service must keep track of half-open connections to complete the connection once the client sends the third step of the handshake. By opening hundreds or thousands of these connections halfway, one can essentially fill up available memory or cause significant slowdown by forcing the system to search through thousands of connection requests with each incoming packet. Another DoS method attempts to fill up a system's hard drive by taking action that causes the system to write to its logs on disk. Many systems will hang or crash if the hard drive fills up. [Northcutt01]

2.6.2.2 Bandwidth Consumption. Bandwidth consumption DoS attempts to fill the network pipe with garbage packets, preventing valid packets from getting through or causing significant slowdown in response to customers. With the dramatic increase in network pipe size, filling these pipes has become difficult. Many hackers install DoS daemons on various unwilling hosts, forcing these hosts to flood a target with far more traffic than a single host could generate. [Northcutt01]

2.6.2.3 Nukers. Nukers use flaws in software to cause a service or the entire OS to crash. Frequently, this attack involves sending a long string that overflows a string buffer. The overflow writes into the code stack, causing the code to crash. Other nukers use flaws in the logic of the code to cause a crash. In many cases, classifying a nuker could be as difficult as buffer overflows since some logic flaws traditionally manifest themselves at the application layer. [Scambray01]

2.6.2.4 Routing and Domain Name Server (DNS) Attacks. Routing attacks involve manipulating the way a router passes packets to their destination. By changing the interface a router uses when sending information along, one could direct packets to an incorrect subnet and cause them to be dropped. The destination host would receive no warning of this attack until all external machines stopped responding. A DNS attack involves changing the hostname to an IP address translation table that the DNS service provides. By directing a hostname to another IP address, one could slow or stop clients from connecting to a particular host. Both of these attacks may be visible to an IDS near the router or DNS server, but would not be visible to the end target of the DoS attack. [Scambray01]

2.6.3 Trojan Horses. A trojan horse is a malicious piece of software often disguised as something useful. By tricking the user into running the software, the trojan horse can install itself onto a machine and perform any number of activities. Many trojan horses provide a back door into a machine, granting the hacker easy access [Northcutt01]. While insertion of the trojan horse may be difficult to detect, use of a back door could be visible to an anomaly detector when, in many cases, a previously unused port suddenly has traffic.

2.6.4 Exploits. An exploit comes in many forms and frequently provides upgraded access for an intruder. Some common exploits take advantage of logic flaws in the DNS daemon BIND or improper access to the windows command line execution tool CMD.EXE [Scambray01]. Detecting an exploit by looking for patterns can be complicated due to the wide range of possible exploits. Many exploits, such as web buffer overflows, may be difficult to distinguish below the application layer. In fact, most exploits use strings or commands given at the application layer, making the detection of anomalies a challenge when only looking at the transport layer or below.

2.7 Summary

This chapter discussed the background and literature on intrusion detection, pattern classification, and network attack categories. The topics introduced are integral for proper design of an attack classifier using pattern-classification tools. Selecting an effective feature set is key to proper classification. Thus, a deep understanding of the data available to a transport-layer anomaly-detector is paramount to classifier success.

III. Methodology

This chapter discusses the methodology used to classify unknown packets into predefined network attack categories. The assumption is that attacks that manifest themselves at the network layer or below will have inherent differences in features between attack classes. These differences will provide a method for properly classifying new packets using a multi-class pattern recognition algorithm. Section 3.1 discusses the overall system design goals for the Network-based Attack Classification System (NACS) pertaining to inputs and expected output, followed in Section 3.2 by a discussion of attack categories. Sections 3.3 through 3.7 discuss the details of the NACS design utilizing the steps outlined in Chapter II.

3.1 System Goals

The NACS is designed to be a post-processing system for an anomaly-detector. While most anomaly-detectors are good at catching novel attacks, they are plagued with frequent false positives. By passing packets flagged by the anomaly-detector through NACS, some false positives may be eliminated. In addition, the system provides attack category classification that could assist system administrators in identifying true attacks.

The classification process has two phases: training and classification. During the training phase, packets of known attacks for each category plus normal packets are provided to the system. The system, with as little human intervention as possible, trains a classifier on all classes to best predict the class of future packets. During the classification phase, unknown packets sent into the system are marked with their classification and given a certainty level indicating the extent to which the system estimates the packet was properly classified.

3.2 Classification Algorithms

The NACS system compares the performance of two classification algorithms. Selection of those algorithms is based on traits of the network data. Since network packets consist of many features, an algorithm that can handle highly dimensional problems is ideal. Normal distribution of the network data cannot be assumed, therefore, algorithms that use the Minimum-Mahanobis distance are not an option. In addition, packet field selection is based on an analysis of the network data and selecting optimal fields is not assured. The ability to determine which fields contribute to either algorithm's success or failure is important. [Duda01]

Taking into account all data requirements, the linear discriminant and radial basis function algorithms were chosen. The linear discriminant algorithm is one of the simplest algorithms and offers ample data on which fields contribute to success and failure. Its ability to generalize is ideal for potentially categorizing unknown attacks. However, it cannot properly train on overly complex patterns. The radial basis function provides a simple and fast method for training on more complex patterns without obfuscating which fields it is using to classify. [Bishop95]

3.3 Attack Categories

The limitations of finding patterns in packet headers must be considered when choosing classes. As discussed in Chapter I, the classifier only examines the fields at and below the transport layer. Therefore, it is not effective at catching most trojan horses or exploits that rely on packet payload rather than packet header information. Some application layer attacks may not be visible below that layer, limiting the ability to detect patterns. For example, the nuke attack category is excluded since most nuke attacks rely on buffer overflows residing in the application layer. While some nuke attacks utilize a large number of faulty packets to crash a system, this strategy may look much like a flood attack using many packets to choke the system or network connection. Data enumeration attacks are frequently used for benign purposes, so distinguishing between malicious and routine data enumeration may prove difficult. In addition, many data enumeration attacks utilize the application layer to transmit the attack data. Categories are chosen based on the literature review and the function of the attack, taking into account the abilities of the pattern classification system. The categories are for normal data, floods and scans.

Some attacks may exist in multiple categories, such as a flooder that also scans for open ports on a system. While some attacks may exist in the center of a class distribution, others may lie on the fringe between two classes. To provide systems administrators with as much information as possible, a future linear discriminant algorithm system must generate a numerical indicator specifying how far from the mean of the distribution the unknown packet lies, given the areas under the probability density. In this research, the Parzen window distribution of the unknown packets is given and extrapolation of the prediction accuracy is left to the analyst based on their risk threshold for false positives and false negatives.

3.4 Sensing

While the sensing phase of many pattern classification applications is often complex, sensing in this application is straightforward. NACS reads in network packet information using the LibPCap C library in either TCPDump raw data file format or directly from a network socket. To limit the scope of the classifier for this research, the sensor is only concerned with reading in TCP packets, and all other packets are ignored. In addition, the only information extracted in the raw network data recording are those fields listed in Table 3.1. While other fields or application-layer information may be useful when looking for patterns in the data, they are ignored in this research to keep the classifier as simple as possible.

Contamination of sensor data is a concern when training the system. The classifier will be improperly trained if attacks are in the wrong categories or mingled with normal traffic during

Field Name	**Range**	**Field Name**	**Range**
IP TOS	0-255	TCP Offset	0-15
IP Len	0-65535	TCP CWR Flag	0-1
IP ID	0-65535	TCP ECN Flag	0-1
IP DoNotFrag Flag	0-1	TCP URG Flag	0-1
IP MoreFrag Flag	0-1	TCP ACK Flag	0-1
IP FragOffset	0-8191	TCP PUSH Flag	0-1
IP TTL	0-255	TCP RESET Flag	0-1
IP Checksum	0-65535	TCP SYN Flag	0-1
TCP Src Port	0-65535	TCP FIN Flag	0-1
TCP Dest Port	0-65535	TCP Window Size	0-65535
TCP Seq Num	0-4294967295	TCP Checksum	0-65535
TCP Ack Num	0-4294967295	TCP Urgent Ptr	0-65535

Table 3.1 IP and TCP Fields Used

training. This research was conducted in a closed lab environment, and both normal and attack data were generated manually to avoid contamination.

For the classifier to properly categorize a packet, it must be different from the other categories in some way. Many attacks form their packets manually (Synpacket, Nmap scans, and Stream1, among others), while normal data packets use kernel interfaces or application programming interfaces (APIs) to generate packets. However, since replies to attack packets are generated with the same APIs or kernel interfaces, it could be difficult to discern replies to attacks from replies to normal packets. For this reason, all traffic is filtered to only allow TCP incoming packets into the classifier. To limit scope and complexity, the system classifies only TCP attacks and normal data.

3.5 Segmentation

Every packet has a definitive start and end; therefore, segmentation is not a concern for this classifier. As stated in Chapter II, while fragmentation can complicate application layer detectors, it does not affect single packet detectors operating below the application layer.

3.6 Feature Extraction

Feature extraction in NACS involves reducing the data and formatting the features for classification [Duda01]. The raw feature data in packets produce poor results when sent directly to the classifier. Fields were carefully selected and modified as necessary for effective classification.

The wide variety of field ranges poses an additional problem. Some fields present a small range from 0-15, while others contain values up to over 4 billion. This disparity causes large fields to have excessive influence over the classification process, where as smaller fields have little impact. Normalizing the data to the same mean and standard deviation helps to ensure that all fields have the same opportunity to influence the classifier. This involves simply subtracting the mean of a field for each data point and dividing the result by the standard deviation of the field.

Problems often arise with linear discriminant analysis when a field contains too little variation after normalization. After normalization, a field may contain all zeros if the value does not change throughout all packets. During the initial testing, it was discovered that removing these unchanging fields removed many singularities.

The goal of the training process is to represent the population of that class of attacks as accurately as possible to best predict future packet classes. Including all packets in the training phase may not be the optimal approach, since, especially with flooding attacks, packets are often identical or close to identical. Ideally, the system should use a data set that best represents the network data category with as few packets as possible. Once all necessary fields are converted or removed, packets found to be identical to other packets are removed. This helps ensure that a smaller number of packets can adequately represent the class of potential future packets, while preventing singularity problems from arising in the algorithm.

Some fields in the packet header are discarded since certain fields could cause the classifier to train on improper data. For instance, if attackers using resource starvation attacks all came from the same IP address, the classifier may train on the address, and the classifier could not properly identify attacks from other IP addresses. To avoid this problem, the IP address is not used in the classifier. Other fields were eliminated if it was unlikely they would change; for example, IP header length is almost always 20 bytes since IP options are rarely used in the data set (The IP version is always four since NACS does not support IP version six).

An important feature of any TCP traffic is the source and destination port number. While all ports are represented on a scale from 1 to 65535, it would be improper to state that port 12 is further away from port 90 than port 50. Each port represents a different service and therefore could be considered nominal data. However, in many cases only the lower 1000 ports are accessed by a packet with the SYN flag set for normal traffic. In other cases, a port scan attack may only scan a certain segment of ports looking for common back-door ports. For these reasons, it may be useful to include port number.

A few fields were modified from their nominal values to either binary or a metric value to better support the pattern recognition algorithms. In some fields, such as the checksum, stating that a particular checksum is higher or lower than another provides no useful information. In this case, checksums are converted to a binary value of 1 if correct and -1 if incorrect.

The TCP sequence and TCP acknowledge numbers help keep track of packet order to ensure that all packets arrive safely at their destination. At the start of a connection, the initial sequence number should be randomly chosen (though this is not the case for all operating systems). The acknowledge number is set with the value of the sequence number in the reply packet. Since these two fields have over 4 billion possible values, the actual value has little meaning and may confuse the pattern classifier.

Useful information can be gained with the relationship of the sequence number to adjacent packets on the network. Many floods and scans have the same sequence number for each packet, while normal data rarely, if ever, has a repeat sequence number. Adjacent normal data packets, in many cases, have sequence numbers close to each other numerically since packets typically increment their sequence number by the number of bytes sent in the packet. Considering these relationships, the TCP sequence number and TCP acknowledge number are replaced with four fields. The first two fields indicate the number of adjacent packets with equal sequence or acknowledge numbers within a set window size before and after the packet. The second two fields indicate the number of adjacent packets with numerically close sequence or acknowledge numbers within the set window size. This can be adjusted to only look at packets after the packet, since unless packets arrive out of sequence, their sequence numbers usually are be slightly higher after the packet being examined. However, for this experimentation, packets are examined before and after to allow for out-of-sequence packets. Which method, if either, provides better results is a topic for future experimentation.

The optimum sequence number window size for proper classification can change depending on the window traffic load. With a higher traffic load, a larger window range may be required. Since the traffic load during the experimentation is relatively low, a small window size of 20 packets before and after the packet is used. In addition, what defines sequence or acknowledge number value proximity must be defined. This value should be chosen based on average packet size, since the sequence number and acknowledge number both increase based on the number of bytes in a packet. Since most packets in the experimentation are relatively small, a proximity value of 20 is used.

Additionally, scans and some floods utilize increasing or random ports for each packet, while normal data often maintains the same port number. Two fields are added that measure how many adjacent packets contain the same source or destination port address. The hypothesis is that normal data has a high number while many attacks have a low number.

Some fields of a data set are thrown out if they lack sufficient variation. In Fisher's linear discriminant function, insufficient variation can cause the inverse of the within-scatter matrix (S_W^{-1}) to approach infinity, causing a singularity. Since many factors may contribute, determining which fields cause the singularity is not straightforward. To determine the 'spread' of a field, the difference between the mean of the two classes is calculated and is divided by the average standard deviation of the two fields so that the highest value indicates the best discrimination for that field. The system can then remove fields by variance until the singularity is eliminated. During algorithm development, it was discovered that removing those fields with a standard deviation of zero usually eliminated the singularity.

3.7 Classification

Once the appropriate features are extracted and normalized, they are presented to the pattern classifier system for training. The training method is different for each pattern recognition technique.

3.7.1 Linear Discriminant Analysis. The space-delimited files produced in the feature extraction phase are read into MATLAB. The discriminator function must discriminate between three different categories, requiring three linear discriminant functions. For each training, two space-delimited files are input into the algorithm. One represents the category being tested while the other represents the 'other' category. This category consists of the other two categories combined. A discriminant vector is calculated that best discriminates the category under analysis from all other categories.

For each classifier, two matrices of equal size are generated and identified as $X1$ for the category under analysis and $X2$ for all other categories. The size used is large enough to represent the range of values in the category under analysis, but small enough so that no duplicate packets exist. During algorithm development, 1000-packet matrices provide good results when compared with larger or smaller matrices. Since no duplicate packets are allowed, increasing the size past 1000 can exhaust the available unique packets. Decreasing the matrix size may cause it to inadequately represent the training data. Since both matrixes must be the same size, the $X2$ matrix is limited to the size of the $X1$ matrix.

The mean for each field in the matrices is calculated, including the combined mean of both matrices. Next, the scatter matrix for $X1$ and $X2$ is calculated as S_1 and S_2, respectively, and is used to determine the within scatter matrix S_W, while the means are used to determine the between scatter matrix S_B. Finally, the eigenvalues and eigenvectors of $S_W^{-1} * S_B$ are calculated. The eigenvector with the highest corresponding eigenvalue is the linear discriminant vector.

Once a discriminant vector is calculated, all unknown packets can be mapped to the vector. A Parzen window is calculated by placing a Gaussian distribution function at each point along the vector, dividing that function by the number of packets, and summing the resulting functions. The resulting probability density is used to define a decision threshold, or a point where everything to the left is one class and everything to the right is another class. This calculation can be performed for each category and, using the results from each category, a conclusion can be drawn. For instance, a packet may go through three classification tests. The first test discriminates between floods and all other categories. The second discriminates between scans and all other categories. Finally, the third discriminates between normal traffic and all other categories. If the packet is classified as a 'flood' on the first, 'other' on the second, and 'other' on the third test, it is probably a flood packet. Likewise, if it is classified as 'other' on the first, a 'scan' on the second, and undetermined on the third, there is still reasonable evidence that the packet is a scan.

3.7.2 Radial Basis Function Analysis. Since the radial basis function algorithm must calculate n coefficients for n equations and n unknowns, where n is the number of packets used to train the classifier, it is desireable to keep the number of packets low to avoid significantly slowing the system. During algorithm development, it was found that 120 packets did not significantly slow the system, yet provided enough variation to effectively train the system. Increasing this value results in a training set that better represents the category, while decreasing it speeds up the algorithm.

A single space-delimited file consists of an equal number of packets for each category. Since only a small number of packets can be used to train the system, packets must be chosen carefully to provide an accurate representation of their class. For the radial basis function algorithm, an equal or close to equal number of packets are chosen from attacks for a specific category. During the testing phase, the flood category used several attacks to train the classifier. If 25 packets are used to train and five attacks are chosen, 5 packets from each attack are combined into a space-delimited file. This file is read into a matrix in MATLAB for processing.

The normalized space-delimited file is read into a MATLAB matrix designated $X_{a,b}$ where a is the number of packets and b is the number of fields. The matrix of Gaussian distributions is calculated using the equations given in Chapter II and given the designator E. The formula for the radial basis function is $y = W * E$. To train the radial basis function, y and E are used to determine W, or the coefficient vector. The vector y_a is a vector of 1s if the corresponding packet E_a comes from the category in question, and a -1 if it does not. For this reason, each category has its own y vector and, therefore, its own W matrix of coefficients.

Once the coefficient vector for each category is discovered, an unknown packet can be sent through the classifier. The formula for categorizing an unknown packet is given in Chapter II. In the formula, the classifier for the first category returns a real value 'e^{c_1}'. This is converted to a probability using: $e^{c_1}/(e^{c_1} + e^{c_2} + e^{c_3} + ... + e^{c_n})$, where c_1 is the category being tested and c_n is the last category. The resulting assurance percentage indicates the category of the unknown packet and gives a numerical indication of the likelihood that answer is correct.

3.8 Post Processing

The post-processing phase of the classification involves simply presenting the data to the user. For each packet, a classification can be provided along with its assurance level. The systems administrator can choose to ignore all floods and scans with high assurance and to concentrate effort on those packets with low assurance or on those that look like possible exploits.

3.9 System Flow

This section outlines the data flow for the preprocessing phase. It describes where linear discriminant and radial basis function analysis differ and where they are identical.

3.9.1 Linear Discriminant Analysis. The process for the linear discriminant analysis is given by the flow chart in Figure 3.1. The diagram indicates the path that network data takes for the

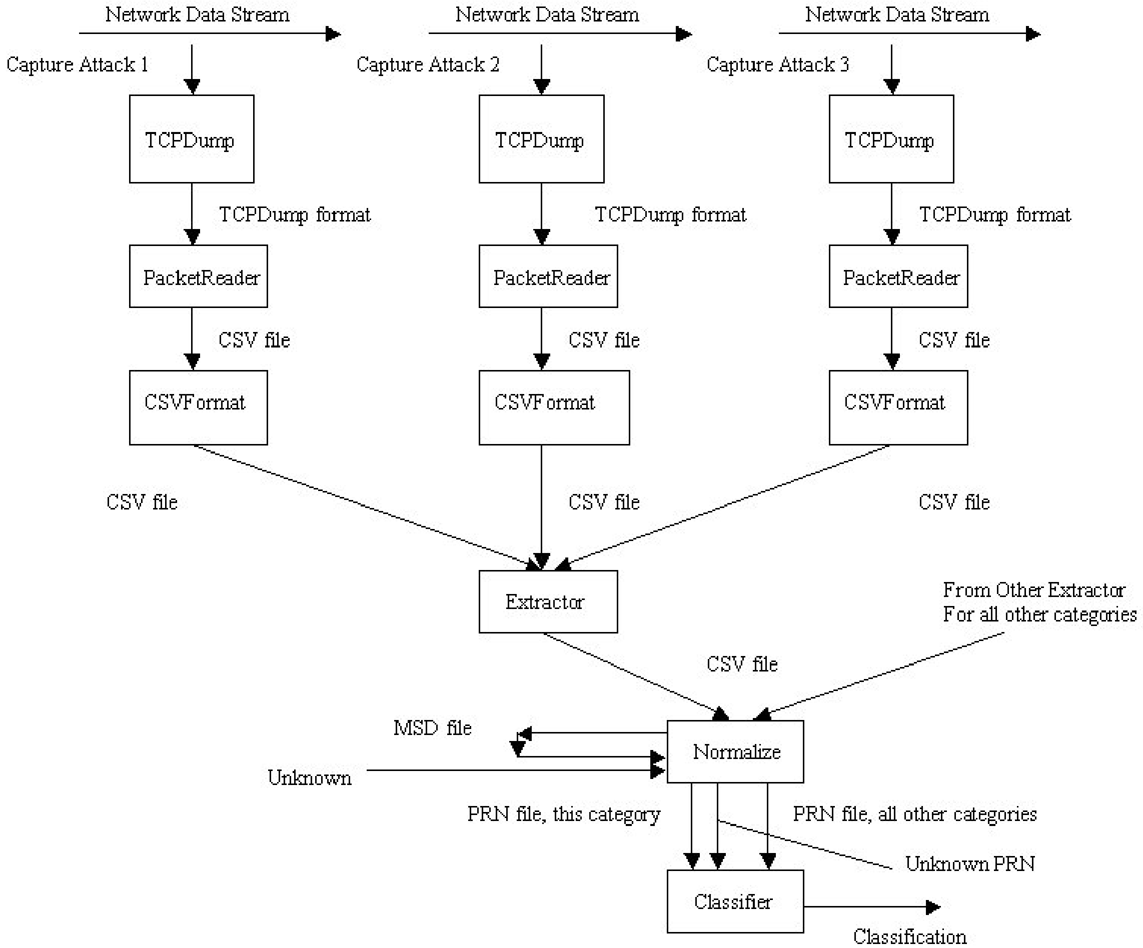


Figure 3.1 The data flow for the linear discriminant algorithm, from sensing to training and finally testing.

training and classification of unknown packets. Attacks are captured off the wire using TCPDump and placed into a TCPDump format file. This file is read in by the Packetreader program, which filters out all but TCP data and converts the checksum values to a 1 for a correct checksum or a -1 for an incorrect checksum. It also converts binary values to -1 for false and a 1 for true. The data is written to a comma-delimited (CSV) file. This file is read into the CSVformat program, where outgoing packets are filtered out. Next, several unnecessary fields are removed, such as the IP address, IPID, and IP Version fields. Derived fields are added to the remaining packets, such as the number of adjacent packets with identical sequence numbers or identical source port numbers. The results are output to another CSV file.

Each attack in a category has its own CSV file generated by the CSVformat program. All these files are input into the Extractor, which adds an equal number of packets (or as equal as possible) from each attack to a single CSV file. This file then represents a category. The resulting category files can be combined using the Unix cat command to create a combined category class, representing the 'all other classes' category used in the linear discriminant function.

Once the CSV files are ready for reading, all are input into the Normalize program. All the data are normalized by the total mean and total standard deviation. It is important that all files used to train the classifier be sent in at the same time, so they will all be normalized with the same mean and standard deviation. The Normalize program outputs an MSD file containing the mean and standard deviation that can be used to normalize future unknown packets. The matrices are each output into a space-delimited PRN file that is ideal for reading into MATLAB.

If the resulting matrices produce a singularity during the training process, the Normalize program is used to eliminate fields of two classes by the ratio of difference of the classes' means and their combined standard deviation. Fields are removed until the singularity problem is resolved. Ideally, this eliminates those fields that contribute little to the classification as their means are too close together or their standard deviations are too spread out. The fields removed are stored in the MSD file so all future packets have those fields removed as well.

Unknown packets are sent through the process much like the training data. They are read in by TCPDump and turned into a CSV file by Packetreader. They are formatted by CSVFormat but skip the Extractor step, moving straight to the Normalize step. The packets are normalized using the MSD file and placed into a PRN file, which can be read into MATLAB and classified with the previously trained classifier.

3.9.2 Radial Basis Analysis. The Radial Basis function classifier preprocessing uses many of the same steps as the Linear Discriminant analysis classifier. The attacks are read into a TCPDump file, converted by Packetreader into a CSV file, and formatted by CSVFormat. The Extractor program combines several attacks into a balanced category CSV file. In the linear discriminant process, each category has its own CSV file. In the radial basis process, all categories are combined into a single CSV file. This file is then read into the Normalize program, which produces a normalized PRN file ideal for training the MATLAB radial basis algorithm. Future unknown packets are sent through the same process as unknowns in the linear discriminant function. They are sent through the Normalize program utilizing a pre-created MSD file and read into MATLAB for classification.

3.10 Summary

This chapter outlines the different pattern classification steps utilized in this research. It reports the data manipulation performed at each step of the preprocessing. It also describes how the actual pattern classification is performed. The next chapter discusses testing and experimentation performed using this system.

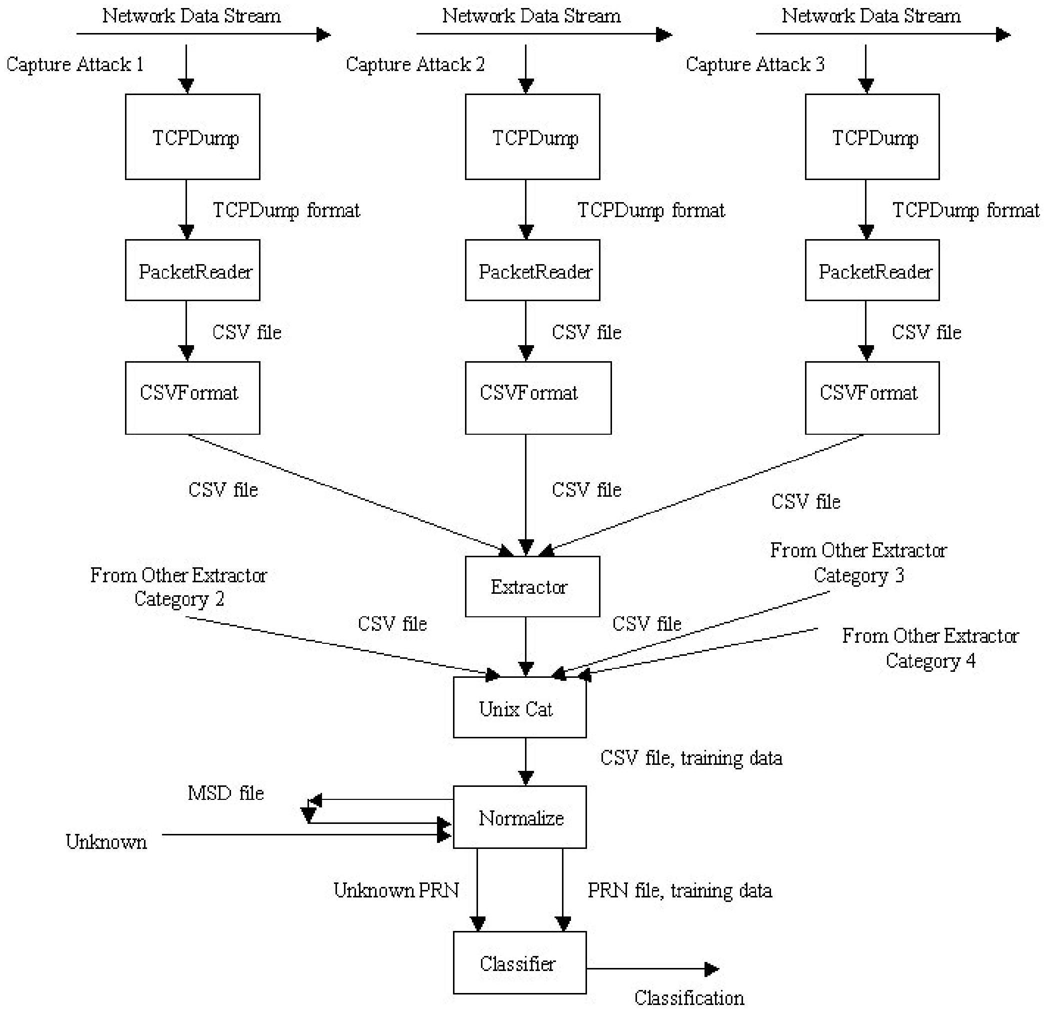


Figure 3.2 The data flow for the radial basis function algorithm, from sensing to training and finally testing.

IV. Test Results And Analysis

The previous chapter outlined the experimental methodology and design. It included a discussion of the various phases of pattern recognition and how each was implemented in the NACS. This chapter presents the detailed plan for testing the NACS pattern recognition system against real attacks and normal data. The testing is designed to determine how well NACS meets the objectives defined in Chapter I. The first objective stated that the system must properly classify attack and non-attack packets using previously trained classifiers. The second objective is that the system automatically determine which features are important. Finally, the third objective is that the system must perform its activities with as little human intervention as possible.

Section 4.1 deals with the specific questions these tests are designed to answer. Section 4.2 covers the test data set design, and Section 4.3 details the experimentatal environment and procedures. Section 4.4 presents and provides analysis of the test results.

4.1 Test Objectives

To properly analyze the first objective outlined in Chapter I, the tests must answer several specific questions:

1. How well does NACS properly classify different samples from attacks it has been trained on?
2. How well does NACS properly classify attacks or normal data it has not seen before?
3. If NACS is wrong, does it adequately indicate unsure results?

These questions can be easily quantified. For the first question, packets are passed through the classifier and the number of correct packets tallied. The second question is answered in a similar manner using packets from an unknown attack. The third question requires more hands-on analysis. Each packet must be examined to determine if it falls close to the decision threshold. Defining 'close' is typically a personal decision for a systems administrator. For testing purposes, close will be within 5% of equal (33%) or at 38% probability. Since for the radial basis function the sum of all classifier outputs must equal 100%, all three categories must be 33.33% to be equal. If the highest value is 38%, then the next smallest value must be 31% or greater, for a maximum 7% spread.

The second and third thesis objectives from Chapter I are difficult to measure quantitatively. This chapter provides a brief qualitative analysis of the NACS system, to include user intervention required, ways this could be automated, and what is difficult to automate and why.

4.2 Test Data Set Design

Testing was accomplished using two sets of data for the linear discriminant analysis and three sets for the radial basis function analysis. The first set of data was used to develop and troubleshoot the algorithms and preprocessing phases. The second set for the radial basis algorithm was used to identify a standard deviation, or sigma value, for the Gaussian distribution that provided the best overall results for that particular data set. This sigma value provides a "best-guess" for maximizing performance on future classifications, where the sigma value could not be tuned since the actual categories may not be known. The final testing set is used on both algorithms to analyze how well the classifier performs.

	Floods			**Scans**	
Attack Name	**Set Left Out**	**Port**	**Attack Name**	**Set Left Out**	**Port**
Httpdflood	1	21	Elitescan	1	Upper Range
Eliteflood	2	80	Nmap_connect	2	Upper Range
Stream1	3	1-2000	Nmap_fin	3	Upper Range
Synk4	4	80	Nmap_syn	4	Lower Range
Synpacket	5	80	Nmap_xmas	5	Upper Range
Hugweb	6	80	Vlad	6	Upper Range

Table 4.1 Attacks used for algorithm development and testing

The first set of six tests consisted of six attacks with a different attack left out of the training data during each test. The attacks used for the system testing and development are listed in Table 4.1. In this and subsequent attack tables, the first column gives the names of the attacks as listed on the Internet. The second column indicates which set the attack was left out of during classifier training. The third column lists the port, or range of ports the attack used. The upper range signifies a range of ports between 1001 and 65535.

In the radial basis algorithm, the attacks listed in Table 4.2 are used to tune the algorithm. In a real-world environment using the NACS system, determining the ideal standard deviation (or sigma) for the Gaussian distribution is difficult given that the actual categories of incoming packets are not known. For this reason, the second set of attacks is used to find the ideal sigma value, anticipating that value will provide a ”best-guess” sigma value for classifying future packets. To better represent a real-world system, the data used to tune the system should be slightly different from the data used to test the system. This choice helps minimize the chance that performance is based on the data chosen, and not on the viability of the algorithm used. To ensure different testing phase data, some attack ports or attack types are changed.

	Floods			**Scans**	
Attack Name	**Set Left Out**	**Port**	**Attack Name**	**Set Left Out**	**Port**
Hugweb	1	80	Nmap_syn	1	Upper Range
Eliteflood	2	110	Ddosscan	2	–
Stream1	3	110	Nmap_xmas	3	Lower Range
Synk4	4	21	Elitescan	4	Lower Range
Synpacket	5	1-1024	Nmap_xmas	5	Upper Range
Httpdflood	6	80	Nmap_null	6	Lower Range

Table 4.2 Attacks used for tuning the radial basis algorithm

Completion of the tuning phase resulting in a sigma value of 0.7. This value was discovered by leaving one attack out and training on the others for six separate tests and indicates the best overall sigma for those tests. This value was used on the third set of attacks to test both the radial basis and linear discriminant algorithms. Since the same set of attacks are used for both algorithms, it is easy to compare performance. Table 4.3 lists the attacks used.

For the radial basis algorithm, an equal number of packets were taken from each attack to total 40 packets per category. The 40 packets add up to 120 packets total, which provided adequate balance during algorithm development between thoroughly representing the data and

	Floods			**Scans**	
Attack Name	**Set Left Out**	**Port**	**Attack Name**	**Set Left Out**	**Port**
Httpdflood	1	80	Nmap_null	1	Lower Range
Eliteflood	2	23	Nmap_connect	2	Lower Range
Stream1	3	Random	Nmap_fin	3	Lower Range
Synk4	4	80	Superscan	4	Upper Range
Synpacket	5	21	Nmap_syn	5	Upper Range
Hugweb	6	80	Elitescan	6	Upper Range

Table 4.3 Attacks used to test linear discriminant and radial basis algorithms

keeping the data set small enough to ensure quick calculation. Packets were taken at even intervals throughout the sample. If the synpacket attack contained 10,000 packets and 8 packets were needed, a sample was taken every 1250 packets. Based on a manual examination of the data set, it appeared that packets cluster at the beginning, middle, and end of a data session or attack. By sampling throughout the data set, this finding increases the variety of packets sampled to better represent the traffic as a whole. If a selected packet is a duplicate of one already chosen, the next packet in the sample is used. If eight packets are needed but only six unique packets are available, a larger sample is taken from the remaining attacks to ensure 40 are chosen for that category. Some attacks vary little from packet to packet, making it difficult to acquire an adequate number of unique packets.

To test each unknown, enough packets are needed to thoroughly test a wide range of packets from that attack. For this research, at most 22 packets are drawn from the attack sample to represent the unknown attack or unknown normal traffic. Increasing this number could provide a wider range of packets to test, however, for some attacks it may prove difficult to find more than 22 unique packets. Since more packets are available for the known traffic, 30 packets were drawn from attacks in a similar manner as the training data to represent a different sample of a known attack. Since the 30 packets were drawn from the same data at a different sampling interval, the data adequately represents similar packets to that on which the data was trained.

To train the linear discriminant function, 1000 packets are drawn in the same manner as the radial basis function for the category being tested. For the 'other' category, 500 packets are drawn from the remaining two categories and combined. The unknown data consists of 100 packets drawn from the unknown attack sample.

4.3 Testing Environment and Procedures

The testing lab consists of five high-end Pentium IV (930 Mhz) machines networked together using a 100MB Ethernet hub and 100MB internal Ethernet cards. The attack machines use Microsoft Windows NT 4.0, Microsoft Windows 2000 and RedHat Linux version 7.2 with the 2.4 kernel. The target machine is a RedHat Linux 7.2 machine using the 2.4 kernel. Data recording is performed on the target machine using TCPDump version 3.4-39 with LibPCAP version 0.4-39.

The pre-formatting phases are accomplished using programs developed specifically for this research. They are written in C++ and are developed on both a RedHat Linux 7.2 system and a Debian Linux v2.2 (Potato release) system. Once formatted, the data is imported into MatLab v5.3.1 R11 running on a Windows 2000 machine where the final analysis is performed and results produced.

The NACS system automates much of the data extraction and normalization required for effective use of the classification algorithms. Duplicate packets are automatically detected and fields with no variation are removed. The data is normalized automatically in preparation for MatLab.

Importing the .csv and .prn files into the various phases listed in Chapter III for the radial basis and linear discriminant algorithms is not automated. Attacks are combined together by hand for each category and imported into MatLab. This action, while effective, can make the classification process cumbersome if the system must be frequently retrained and new data classified. Transporting between phases could be easily automated for future research to minimize the human effort required for the system.

Certain areas of the system are difficult to automate. First, the user must choose attacks on which to train the system. A wide range of attacks must be chosen, but care must be used when choosing attacks to ensure proper classification and avoid blurring the category boundaries. For instance, if a scan is actually an attack but is trained as a scan, the boundary between the classes could be difficult to discern. Second, transporting the data between the C++ programs and the MatLab program must be done by hand. There are some C++ libraries that perform the required MatLab functions and could help automate this step, but writing the program would require many man-hours. It could, however, be valuable for future research or a production system.

4.4 Test Variables

Throughout the previous chapters, several variables were discussed whose values were chosen based on prior testing or observations. These variables are listed in Table 5 with a short description, the section in which it is defined, and the value chosen. While care was taken to choose values that provide optimum results, adjusted values may produce better results.

Description	**Value Used**
Packet window size for derived fields	20 packets
Proximity for TCP Sequence and Acknowledge numbers	20
Number of packets per category for training the linear discriminants	1000 packet
Number of packets per category for training the radial basis function	40 packets
Number of packets per unknown category during the radial basis tests	22 packets
Number of packets per sample of known category traffic during the radial basis tests	30 packets
Unknown percentage cutoff point	38%

Table 4.4 Attacks used to test linear discriminant and radial basis algorithms

4.5 Test Results

The tests described in previous sections produced the results outlined in this section. Complete results can be found in Appendices A and B for the linear discriminant algorithm and Appendix C for the radial basis algorithm.

4.5.1 Linear Discriminant Algorithm Results. Previous research [Noel01] included in Appendix D, demonstrated that in some cases, attack traffic and normal traffic for a specific port can be divided effectively by Fisher's linear discriminant. The NACS system tests Fisher's linear discriminant against a much wider range of traffic. Multiple ports, attacks, and traffic types were used to train and test the linear discriminant algorithm with mixed results. The first test analyzes whether the linear discriminant algorithm can correctly categorize the Httpdflood as a flood and the Nmap NULL scan as a scan. The classifier is first trained on five other attacks per category. Httpdflood is tested against three linear discriminant vectors. The first, shown in Figure 4.1, indicates the distribution of the flood category including five flood attacks. This category is compared with the combination of the other two categories. The figure indicates that the majority of the packets fall on the left side, or in the flood category. Ideally, all packets should fall to the left of a decision threshold defined by the systems administrator. In this case, the decision threshold chosen is indicated by the dotted line down the middle of the graph. Most flood packets fall to the left of the line while most "other" packets fall to the right. For the Httpdflood packets, most fall to the left of the line, but some fall to the right. This provides evidence that most packets come from a flood, which in this case would be correct.

While promising, this linear discriminant algorithm is not a perfect classifier because almost one-third of the Httpdflood packets are improperly classified or marked as unknown. Figure 4.2 shows the same unknown flood when compared with the scan category. In this case, all packets fall where they should in the "Other" category. Taking both graphs into account, one could correctly surmise that the packets were either flood packets or normal packets, depending on where the packet fell in the first graph. If the third graph indicates that the unknown packets are in the "Other" category, those would be more evidence that the unknown packets are a flood.

Figure 4.3 does not provide the evidence needed. Judging by the lack of discrimination in Figure 4.3, it appears there is very little difference between the normal data and the other two

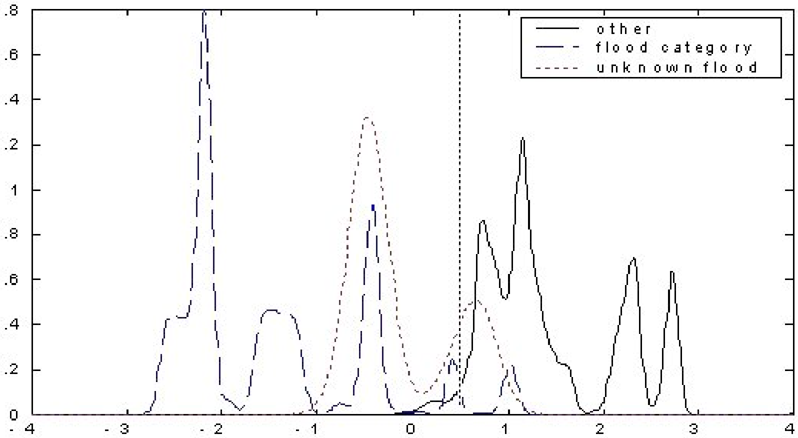


Figure 4.1 Httpdflood compared against "flood" and "unknown" categories
The unknown is mapped to a linear discriminant trained with a "known floods" category and an "all other" category and displayed using a Parzen window distribution. The dotted line down the middle indicates a possible decision threshold.

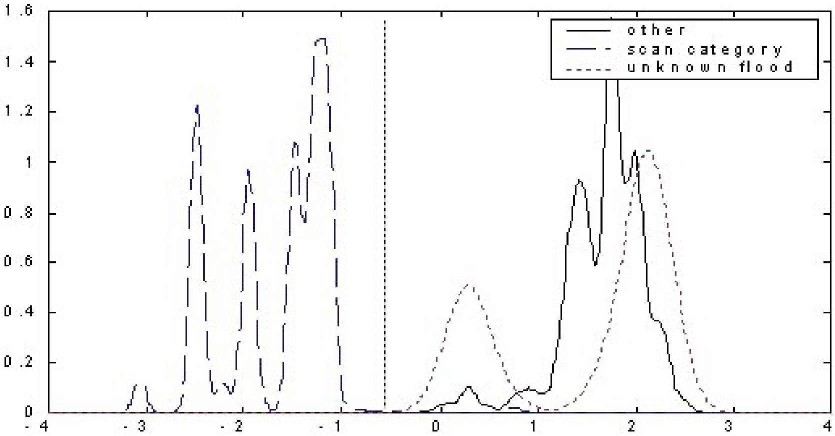


Figure 4.2 Httpdflood compared against "scan" and "unknown" categories
The unknown is mapped to a linear discriminant trained with a "known scans" category and an "all other" category and displayed using a Parzen window distribution. The dotted line down the middle indicates a possible decision threshold.

categories combined. It is difficult, given the graph in Figure 4.3, to discern where one category starts on the discriminant vector and another ends. Where the unknown flood peaks, both the normal category and the other category peak as well, making it impossible to make an educated guess.

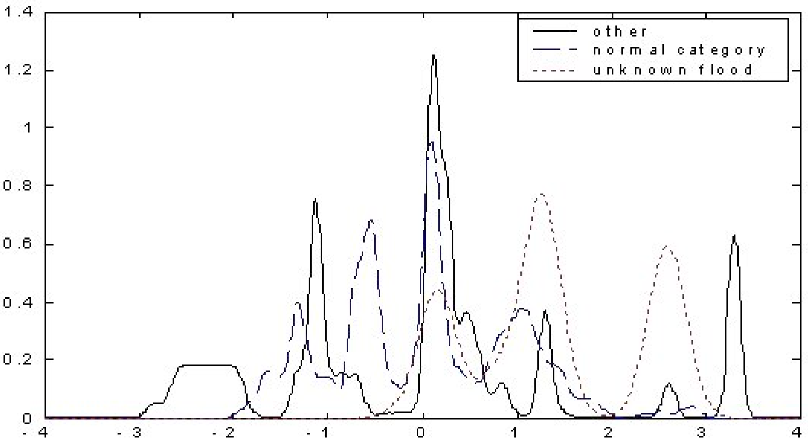


Figure 4.3 Httpdflood compared against "normal" and "unknown" categories
The unknown is mapped to a linear discriminant trained with a "known normal" category and an "all other" category and displayed using a Parzen window distribution. The poor discrimination between the two categories makes it impossible to establish a decision threshold.

Evidence seems to support the Httpdflood belonging to the flood category, given the first two graphs. While the third graph comparing the 'normal' category against the 'other' category produced poor results, the first two indicate it is a possible flood and not a scan. A properly discriminating third graph would have helped validate these results. However, a correct conclusion can be drawn from available data that the Httpdflood is in fact a flood.

Testing the lower range NMap NULL scan from the first test against the known flood and scan categories produced better results, as indicated in Appendix A. However, as with Test 1 shown in Appendix A.2, comparing the normal category against the other two categories consistently failed to define a clear line of discrimination between the two categories, as indicated by the graphs in Appendix A. It is interesting to note that when normal unknown data is passed through a classifier trained with a "normal" and "other" category, the unknown normal packet distribution produces nearly the same shape as the normal category, as demonstrated in Appendix A.3 given the positions of the various peaks.

One hypothesis on why the normal category graphs are poor discriminators is that normal traffic is much more varied than scans or floods. However, combining scans and floods produces

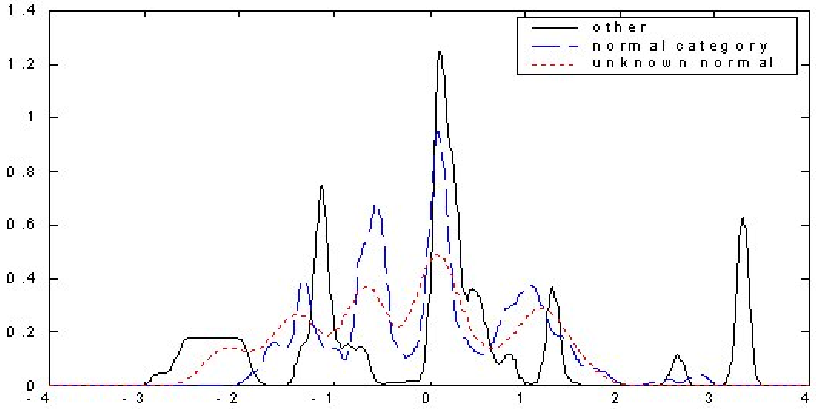


Figure 4.4 Normal traffic compared against “normal” and “unknown” categories
The normal unknown traffic is mapped to a linear discriminant trained with a “known normal” category and an “all other” category and displayed using a Parzen window distribution. The poor discrimination between the two categories makes it impossible to establish a decision threshold.

a more varied pool of packets and, therefore, makes it harder to distinguish between normal and attack categories.

In some tests, classification became impossible due to poor training. Since for each test, two attacks were not trained on so they could be used as the unknown, this poor discrimination may indicate dependance upon the attack left out. In Test 3, when the Stream1 flood and the Nmap FIN scan is left out, the linear discriminant algorithm has difficulty properly discriminating. One possible contributing factor involves the ports the other floods attacked. The Stream1 flood was the only multi-port flood used. When it was left out, the attacks hit only ports 21, 23, and 80, limiting the variety of traffic. In addition, two lower range scans would also have packets directed to those ports, making discriminating difficult.

Tests 4 and 5 generated poor discriminators. However, discovering the reason for the poor performance is not straightforward. Unlike Test 3, both Test 4 and 5 leave out floods that strike only a single port. Given that a pattern may exist across multiple dimensions, discovering a connection between a field and poor performance can be difficult. In Test 5, the good discrimination when comparing the flood against the “Other” category, while the poor discrimination when comparing the scan against the other category may indicate that the NMap SYN scan left out is essential for distinguishing between scans and the “Other” categories.

While these graphs have potential for helping decide the category of a packet, they are limited since they do not give a clear answer. In some cases, the distinction is clear since the categories are well-defined. However, as the figures in Appendix A demonstrate, sometimes the categories are difficult to discern. For example, the Test 2 Port 23 Eliteflood in the Appendix A.4 test against the flood category appears to indicate the Eliteflood is neither a flood nor a scan. The Test 2 lower range Nmap Connect scan in Appendix A.5 appears to indicate it is both a flood and a scan. The poor discrimination in the Test 3 Stream1 flood in Appendix A.7 tested against the flood category make it impossible to determine a category.

Where there is a distinct decision threshold and where the majority of the unknown packets fall on one side of the decision threshold, a decision can be made on how to classify the unknown packets. This produces the results listed in Table 4.2. The cause of most incorrect classifications appears to be a lack of variation between classes. If the two classes are similar, then Fisher’s linear discriminant has trouble finding a vector that discriminates between the two classes. This makes classification of future unknown packets difficult.

		Summary	
	Flood Correct	**Scan Correct**	**Normal Correct**
Test 1	Y	Y	Y
Test 2	N	N	Y
Test 3	Y	Y	N
Test 4	N	N	Y
Test 5	N	N	Y
Test 6	Y	Y	Y
Total Correct	**3**	**3**	**5**

Table 4.5 Linear discriminant analysis results of the various graphs

Unlike the previous research on using linear discriminants to classify network traffic [Noel01], this application does not perform exceedingly well. Both floods and scans have approximately a 50% success rate. While this is 150% better than a random guess yielding a theoretical 33%

success rate, it still gets half of the packets wrong. This may be due to several factors, from far less homogeneity in both attacks and normal traffic to using three categories instead of two. Combining multiple categories into the "Other" category may make the data too heterogeneous to effectively discriminate between the category being tested and the "Other" category. The category patterns may be too complex for a simple linear discriminant and may require a pattern recognition algorithm that can learn complex patterns.

4.5.2 Radial Basis Algorithm Results. The radial basis function pattern recognition algorithm is better suited to handle complex patterns. This means the classifier may be able to accurately train on known data and maintain a high success rate when classifying data from attacks it has seen before. It also means it may not generalize well enough to properly classify attacks it has not yet seen.

The first set of tests was used to discover the best standard deviation, or sigma, of the Gaussian distribution that produced the best results. The results of this test are listed in the first graph on Appendix C. The algorithm produces three percentage values for each unknown packet, one value for each category. The category with the highest percentage value indicates the expected category of the unknown packet. The percentages must add up to 100%, thus if the highest value is close to 33%, a lack of certainty is indicated. For this experiment, if the highest probability level was within 5% of 33%, or 38%, then it was counted as an unknown even if the answer was correct.

The tuning phase produced fair results when trying to detect unknown floods, resulting in a 53% detection rate. It performed much better in detecting unknown scans and normal data, with 94.7% correct for both.

When detecting a new sample of a known attack, the tuning phase performed very well, correctly classifying 98.9% of scans and 100% of flood packets. Since there is no difference between normal unknown and what would be normal data it has trained on, there is no 'Trained Normal' category.

The Tuning phase had some trouble on unknown packets, as one might expect given that some packets look normal. In many cases, where it made a mistake was on an attack that was designed to flood the target with valid connections, tricking it into using up resources trying to keep track of all the connections. These packets would be tough to distinguish from normal traffic as they are designed to look normal. This is confirmed by the classifier's poor results.

The final radial basis test produced the results found in the second and fourth tables of Appendix C. The first test performed very well, only missing two normal packets. However, while packets marked unknown are preferable to incorrectly marked packets, the two errors were marked incorrectly.

Subsequent tests performed with mixed results. Test 2 did not perform very well, incorrectly classifying 18 flood packets and marking the remaining four as unknown. Scans performed slightly better, only marking 7 incorrectly but marking the remaining 15 as unknown. Test 3 and 4 marked all 22 flood packets as unknown, which provides better information to the administrator than marking them incorrectly.

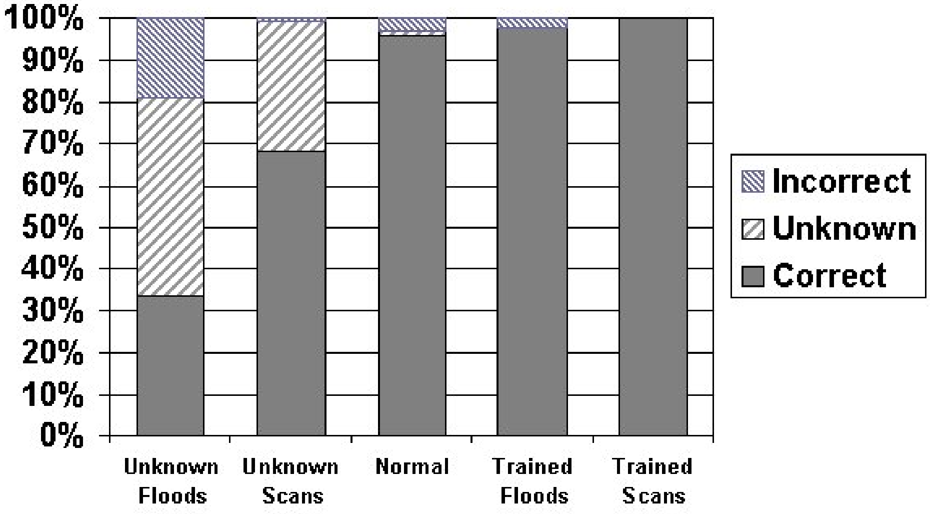


Figure 4.5 Radial basis test phase overall classification results for unknown packets.

The resulting probabilities for each category are given in Figure 4.6. The closer a packet is to 33%, the more likely the packet will be classified as unknown. The closer a packet is to 100%, the more probable that packet is of the class indicated. Figure 4.6 shows the Httpdflood attack packets plotted by their probability. In this situation, all packets were properly classified, however, a few packets came close to being classified as unknown.

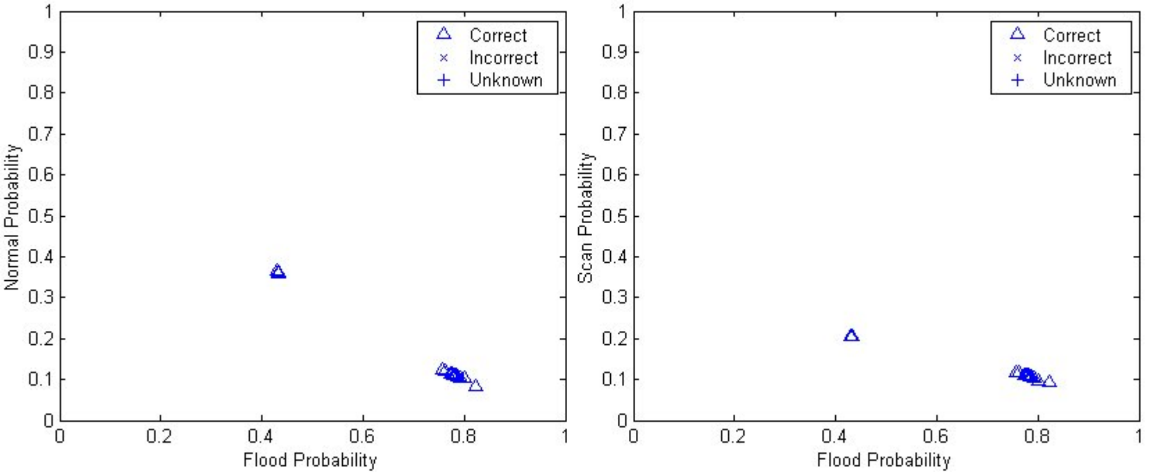


Figure 4.6 Probability results of the radial basis function algorithm for all three categories using Httpdflood packets as the unknown

Not all packets were classified properly, as the combined graphs in Figure 4.7 indicate. In the left graph, all but the bottom four packets were marked as incorrect. The last four packets in the fairly linear sequence of packets were marked as unknown. Contrast this graph with the Httpdflood packets in Figure 4.6. While the classifier categorized most of the Eliteflood packets incorrectly, it was closer to equal than the Httpdflood packets in which most were around 80% for the flood category.

The normal traffic classified in Figure 4.8 appears more varied than the tightly clustered attacks in Figures 4.6 and 4.7. This corresponds with the results of the linear discriminant analysis of the same data in Appendix A.

Test 5 presents some interesting results illustrated in Figures 4.9 and 4.10. Both the SynPacket flood unknown and Nmap SYN scan unknown are clustered relatively close to 33%. This could

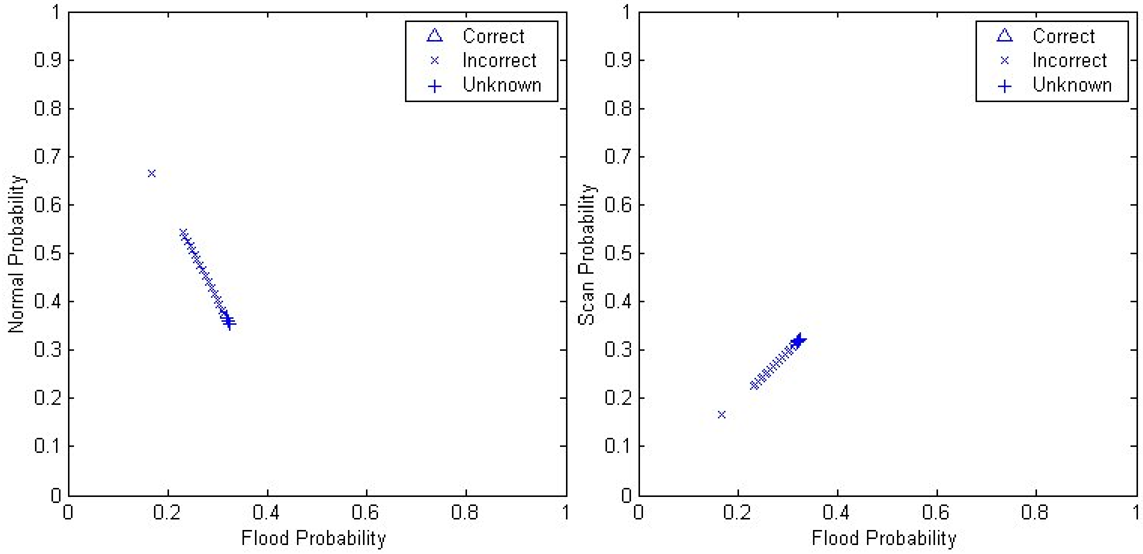


Figure 4.7 Probability results of the radial basis function algorithm for all three categories using Eliteflood packets as the unknown.

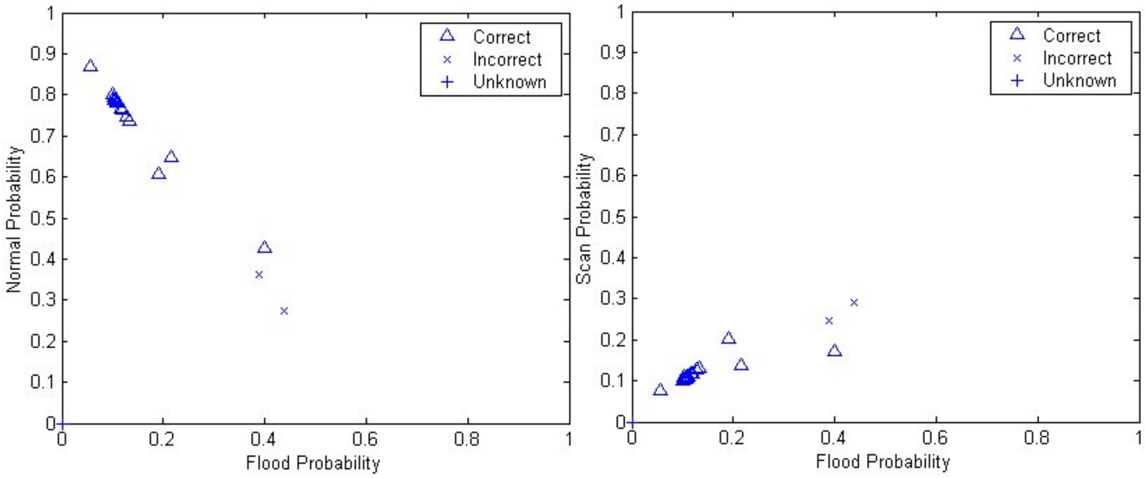


Figure 4.8 Probability results of the radial basis function algorithm for all three categories using normal packets as the unknown.

indicate lack of proper training at the particular position on the data landscape on which these unknowns normally reside. Given that both utilize the SYN TCP flag to perform their tasks, this could indicate a lack of similar SYN packets in the training set.

Overall, the classifier works very well when classifying attacks it has seen before, with a 97.8% success rate for floods and a 100% success rate for scans. Since the large majority of

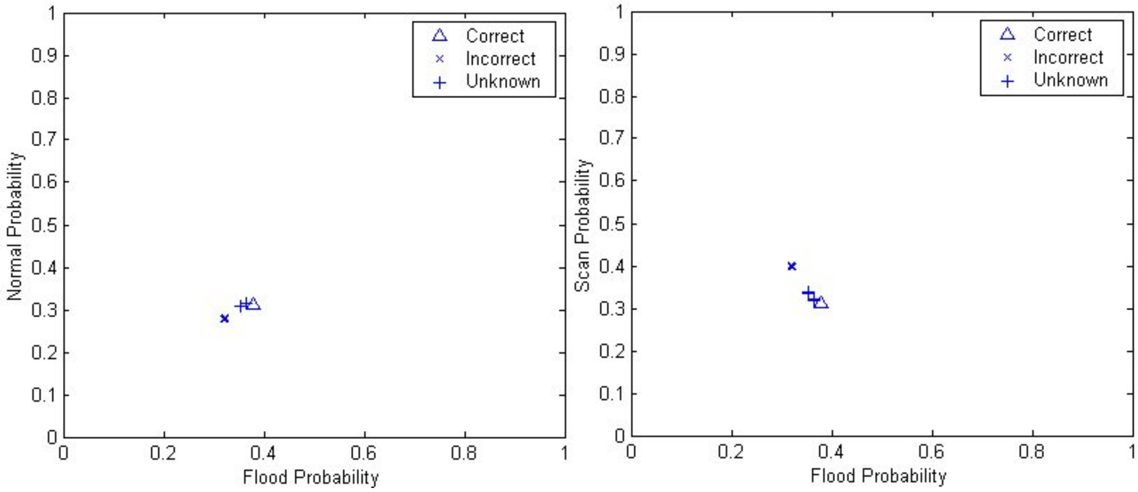


Figure 4.9 Probability results of the radial basis function algorithm for all three categories using SynPacket packets as the unknown.

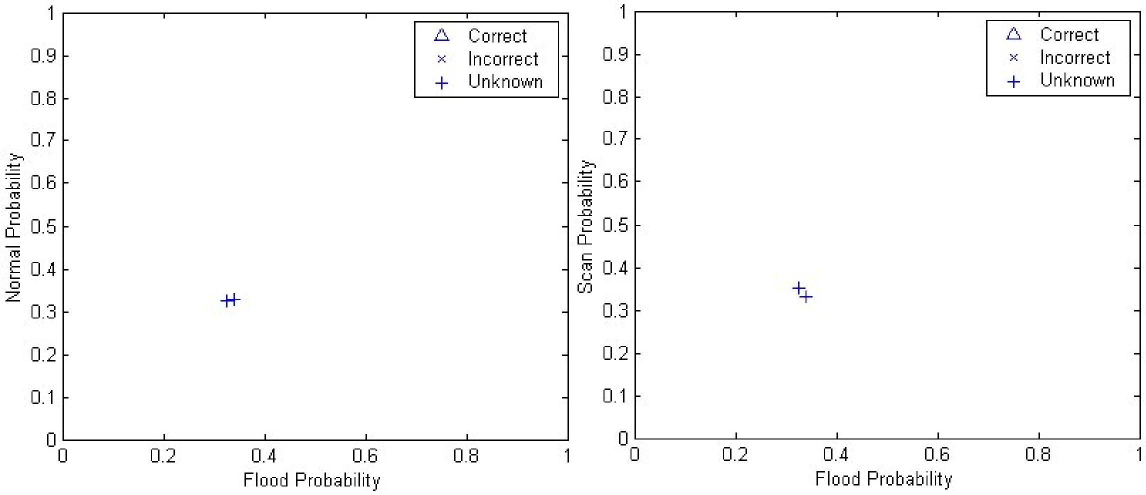


Figure 4.10 Probability results of the radial basis function algorithm for all three categories using Nmap SYN packets as the unknown.

attacks come from novice hackers running pre-made scripts, this is ideal. The few unknown attacks the system comes across have a 19% chance of being misclassified for floods and a 1% chance of being misclassified for scans. A large majority of unknown floods (48%) would be marked as unknown, requiring further research by the systems administrator. However, 33% would be correctly identified, eliminating a full third of the packets a systems administrator would need to manually handle. Scans perform better, correctly classifying 65% of all packets, while unknown normal traffic is correctly classified 96% of the time.

Decreasing the range of uncertainty improves the number of correct classifications. If the results are calculated with little or no range of uncertainty (i.e. anything but 33% is not considered an unknown) , the percentage of correctly classified floods rises from 33% to 57%. However, this produces 43% incorrect classification, which would make the system too unreliable for most systems administrators when classifying floods.

4.5.3 Algorithm Latency. The speed of each algorithm depends on a variety of factors. Since MatLab is an interpreted language, it runs slower than if the system were coded in a compiled language. For a real-world system, the classifier should ideally be written in a compiled language such as C++. In addition, many aspects of the MatLab code could be optimized for better performance.

Each algorithm was run on a 930Mhz Pentium IV machine. The linear discriminant algorithm can map 100 unknown packets down to the discriminant vector in less than a second. However, generating the Parzen window used to display the data can take 10-15 seconds. The radial basis function algorithm calculates approximately 20 packet category probabilities per second.

While these results are obtained too slowly for a real-time sensor, they are adequate for the intended purpose. The packets processed by NACS would be those presented by a standard anomaly detector. The classifier would categorize the anomalous packets to help minimize the number of packets a systems administrator would need to address.

4.6 Summary

The results indicate that, while pattern recognition is a useful tool for identifying patterns in packet headers, it is not a “silver bullet” for detecting or classifying unknown attacks. These results demonstrate that it is useful in classifying attacks it has seen before based merely on packet header information and that it can help decrease the workload of an administrator by eliminating some of the packets from previously unknown attacks. This finding could be useful if an administrator is swamped with attacks, both known and unknown, in an attempt to hide dangerous attacks in a sea of IDS warnings. Eliminating many of the attacks using a pattern classifier could help detect the dangerous attacks.

V. Conclusions and Recommendations

The previous chapters present evidence indicating that pattern classification may be a viable tool for use on packet header information. This chapter briefly analyzes the results described in Chapter IV to include potential impact. It concludes with a discussion of areas for future research.

5.1 Final Analysis

The goal of this research was to test how well two pattern classification algorithms accomplish the three objectives outlined in Chapter I.

- **Objective 1:** Develop pattern classification algorithms that indicate the category of an attack for unknown packets.
- **Objective 2:** Determine automatically which features are most important for discriminating between attacks.
- **Objective 3:** Require little to no human intervention after the training process to properly classify attacks.

The pattern classification algorithms used meet Objective 2 by design. Fisher's linear discriminant vector is chosen in such a way as to capitalize on those features that best discriminate between the two categories during training. The radial basis function selects coefficients for the W vector that maximize the impact of certain features and minimize the impact of others. Choosing these fields requires no interaction with the user, aside from the feature extraction phase.

NACS meets Objective 3 to an extent. Ideally, a real-world system using the process outlined in this research would eliminate several of the steps that require human intervention. As it stands, human intervention is required mainly to pass information between the different processes in NACS, which would be cumbersome for a real-world system. However, the data extraction and normalization is performed automatically.

The first objective is the real measure of the success of NACS. The first algorithm, linear discriminant analysis, was accurate in slightly more than half of the samples provided. While more accurate than a random guess when presented with an unknown attack, it did not meet Objective 1 adequately.

The radial basis function performed better when given an unknown packet, marking it correctly or as an unknown in the large majority of cases. When presented with a different sample of an attack it had been trained on, it performed with almost 100% accuracy.

The results indicate that, while patterns in packet headers do exist, they are fairly complex, limiting the capability of the linear discriminant algorithm. The better-suited radial basis function algorithm is able to train on the pattern and use the pattern to predict the category of future packets. If the radial basis function cannot determine a category with enough accuracy, it is indicated in the probabilities returned by the algorithm.

This system is not designed to be a front-line sensor. It is best suited as a post-processing filter for an anomaly detector. As discussed in Chapter II, one of the biggest drawbacks to an anomaly detector is the large number of false alarms. While it will catch novel attacks, this drawback prevents widespread use. Being able to filter those alarms could ease the administrative load required to discern the benign from truly dangerous attacks.

5.2 Future Research

Pattern classification algorithms are relatively new tools when used to classify attacks. While promising, there is still much research needed for them to be viable tools. The pattern classification area of research is rapidly expanding, especially in the area of intrusion detection, presenting a ripe field for new research.

5.2.1 UDP and ICMP Classifiers. This research concentrated on only TCP attacks. This leaves out several attacks that consist of mainly UDP or ICMP packets. A classifier could be developed to determine if the same methods could be used to categorize UDP or ICMP packets. In addition, the possibility of including other categories could be addressed.

5.2.2 Limiting Data Scope. As discovered during algorithm development, the more complex a data set, the greater difficulty pattern recognition tools have in discerning patterns. During previous research [Noel01], the linear discriminant provided effective discrimination between attack and normal data on a single port. However, the multiple category, multiple port traffic in this research proved more difficult to discriminate.

Better discrimination may be achieved by decreasing the data scope. For instance, a separately trained classifier could listen at each common data port. When new data arrives, it is passed to a classifier based on its port number. This could help the classifier by limiting the variety of traffic in each category and making it easier to discern patterns.

The classifier could also be limited to other fields not used in this research. A classifier could detect patterns in IP addresses under the premise that certain subnets will produce a larger number of hacker attempts. The limited scope classifier outputs could become inputs into another classifier to provide for a wider assortment of fields and to detect complex patterns.

5.2.3 Patterns In Data Sessions. This research uses only packet header information to detect attacks. However, this single packet approach may miss certain patterns that exist across multiple packets. A session-based classifier could detect patterns that the single-packet classifier may miss. A session is usually a collection of packets linked by some common purpose and often common fields. Transferring a graphics file may consist of many packets but a single session. A telnet connection also consists of many packets, but usually only one session. Utilizing fields derived from the session in a pattern classification tool could yield useful results. These fields may consist of duration, data size sent, data type, and number of packets sent.

5.2.4 Payload Pattern Classification. In addition to utilizing pattern classification on packet header information, pattern classification may have uses in the packet payload. Due to the wide variety of possible traffic, this area is far more complicated than using header information. In many cases, one will not know whether to expect standard text, graphics files, binary executables, or even an encrypted payload. Extracting useful features from an unstructured data stream would be difficult, if not impossible. However, this does not preclude using pattern classification on payloads that already have an established structure or can be broken up into basic elements.

5.3 Conclusion

Overall, this research provides evidence that pattern recognition algorithms capable of training on complex patterns, such as the radial basis function algorithms, may provide methods for classifying packets and thus help filter noise from attacks worthy of attention. While there is still much research that needs to be done to develop a viable system, the initial results show promise. While pattern classification still has far to go to surpass the human mind, the ability of computers to handle many dimensions can be essential for limiting the data load and dimensions the human mind must process.

Appendix A. Linear Discriminant Graph Results

The following results were produced during the final testing of the linear discriminant algorithm. Five attacks were used to train the algorithm while the remaining attack was sent through as the unknown attack. The results were placed into a Parzen window Gaussian distribution. For each unknown attack, three graphs were produced. The first shows the unknown packets mapped onto a line discriminating floods from all other traffic. The second discriminates scans from all others and the third discriminates normal traffic from all other traffic.

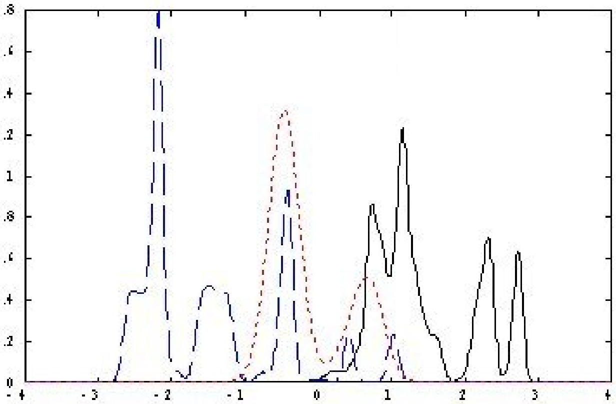
Each page of this Appendix contains three graphs that correspond with the following three legends. The ‘other category’ is a combination of the two remaining categories. The actual ‘unknown traffic’ used is given at the top of the page.

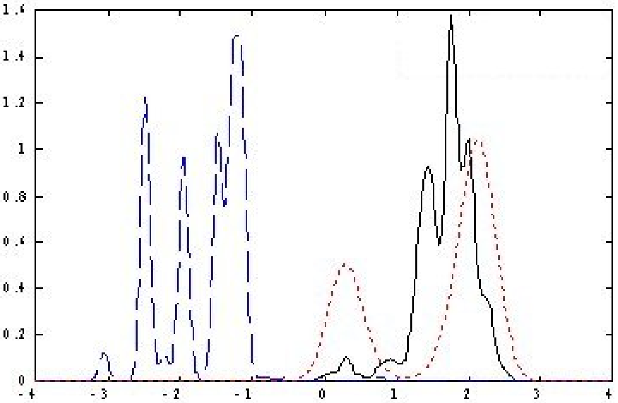
——— Other Category
— — · Flood Category
- - - - Unknown Traffic

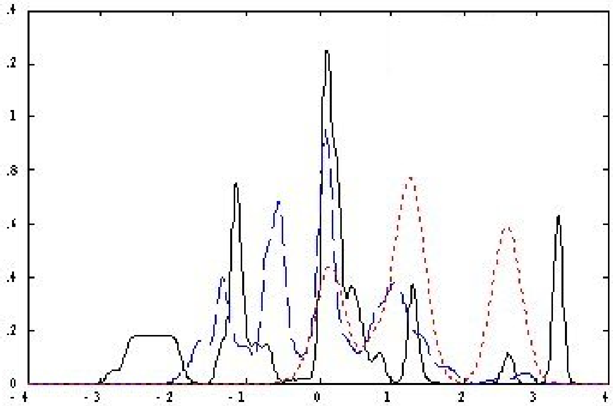
——— Other Category
— — · Scan Category
- - - - Unknown Traffic

——— Other Category
— — · Normal Category
- - - - Unknown Traffic

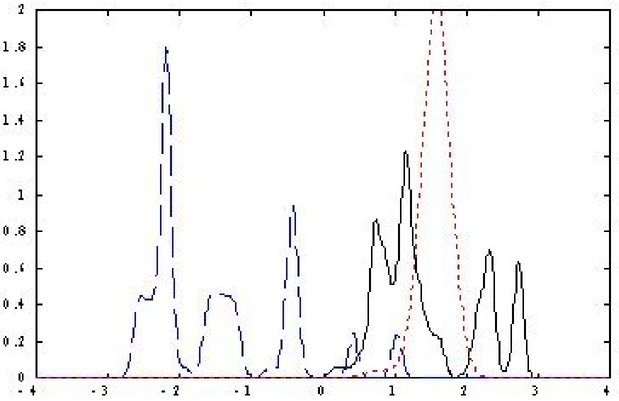
A.1 Test 1: Httpdflood (Flood Category) as Unknown

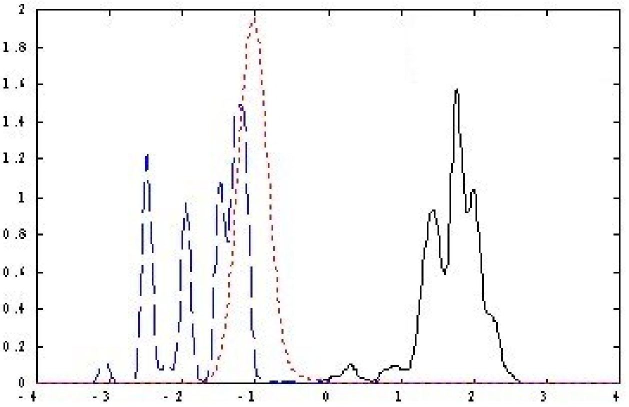


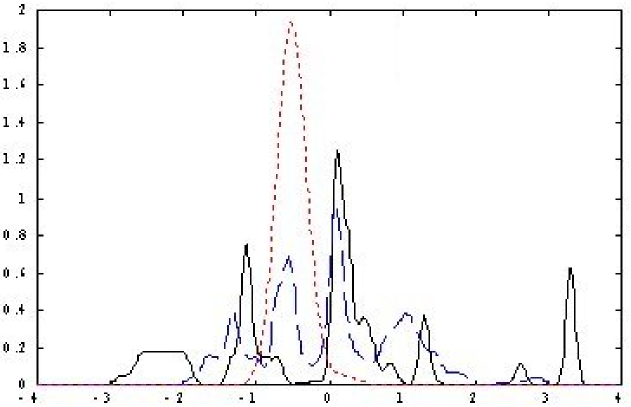




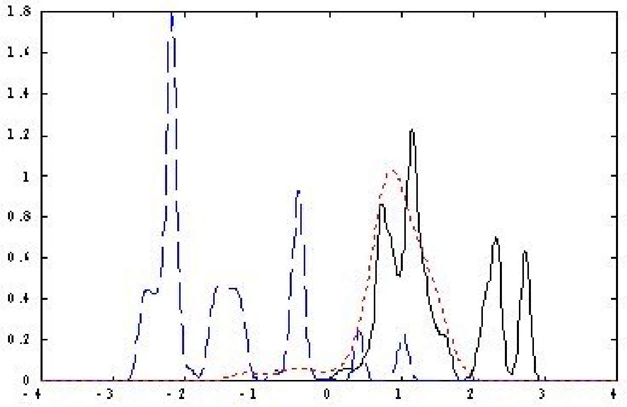
A.2 Test 1: Lower Range Nmap NULL (Scan Category) as Unknown

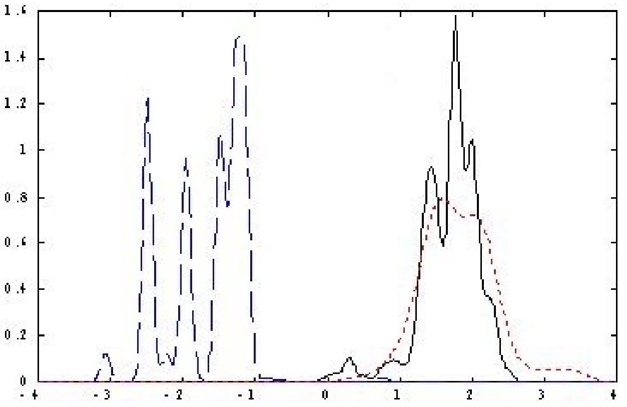


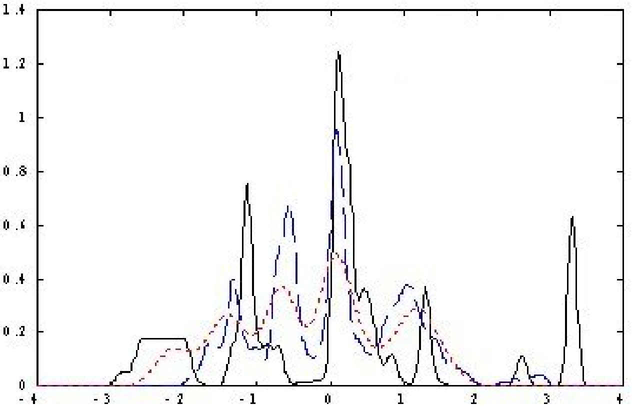




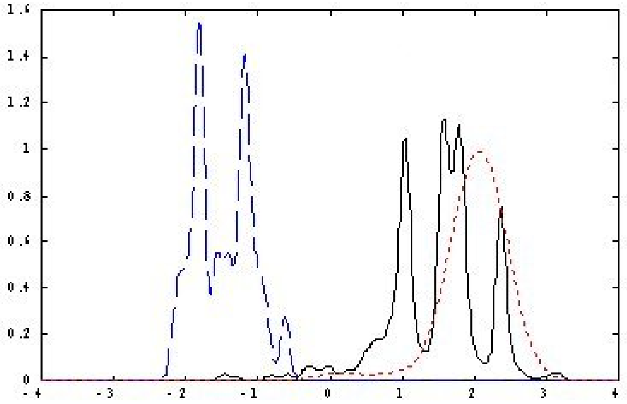
A.3 Test 1: Normal as Unknown

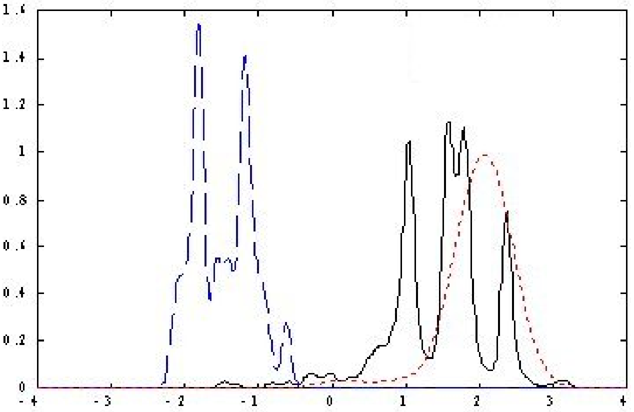


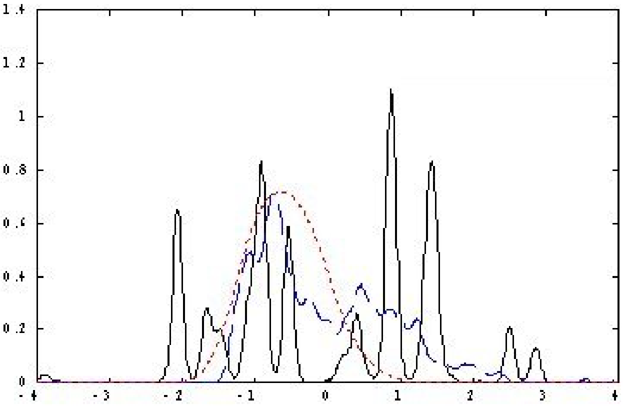




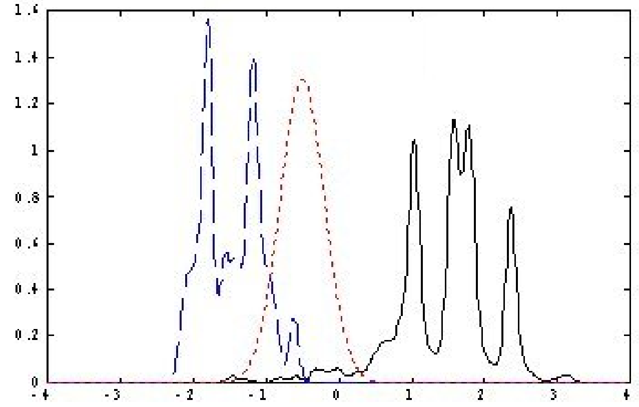
A.4 Test 2: Port 23 Eliteflood (Flood Category) as Unknown

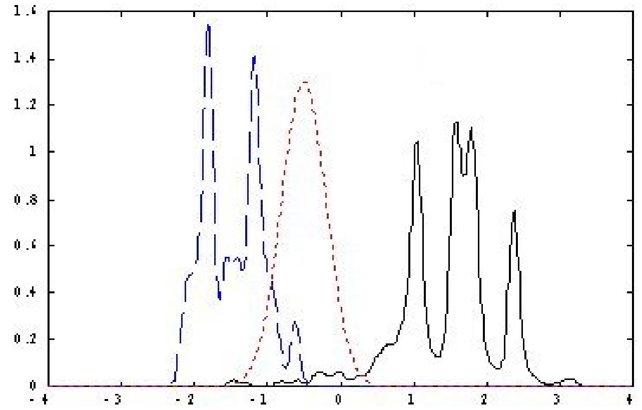


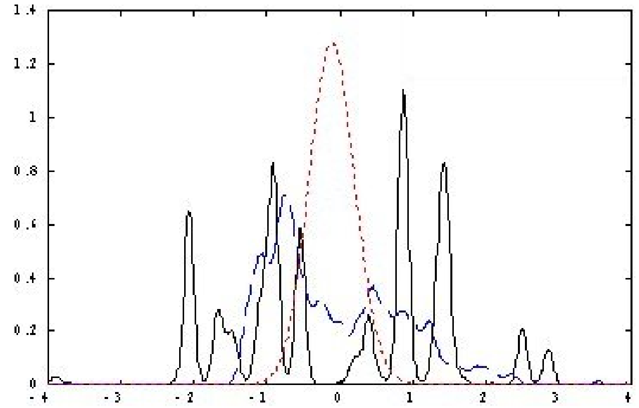




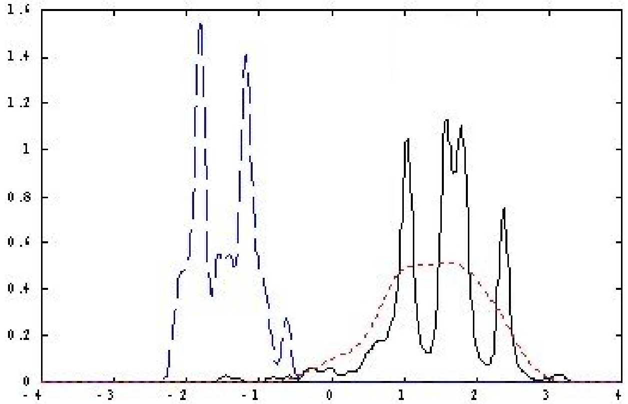
A.5 Test 2: Lower Range Nmap Connect (Scan Category) as Unknown

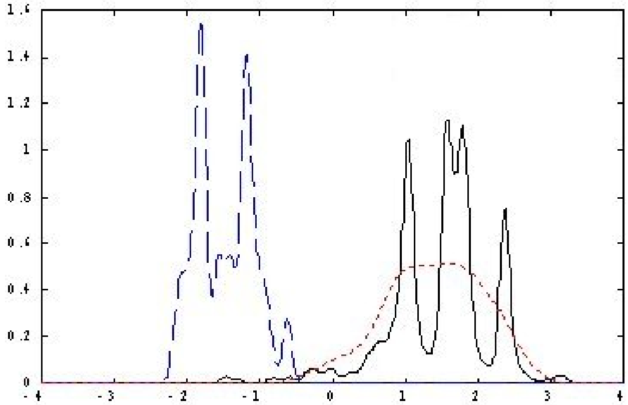


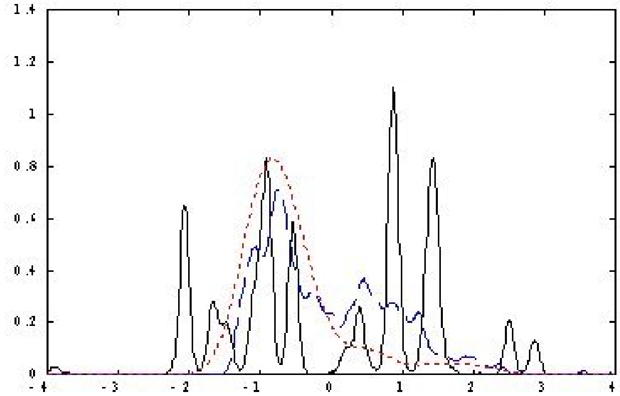




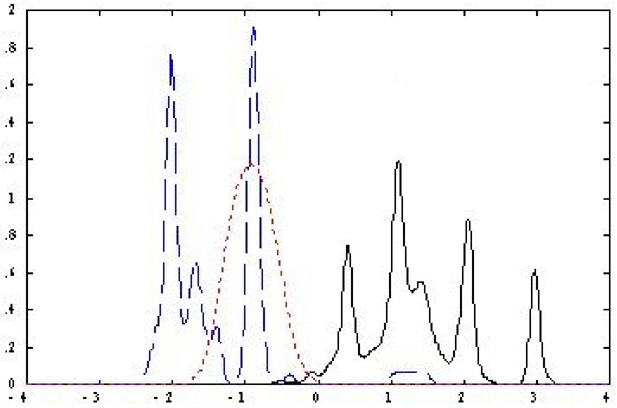
A.6 Test 2: Normal as Unknown

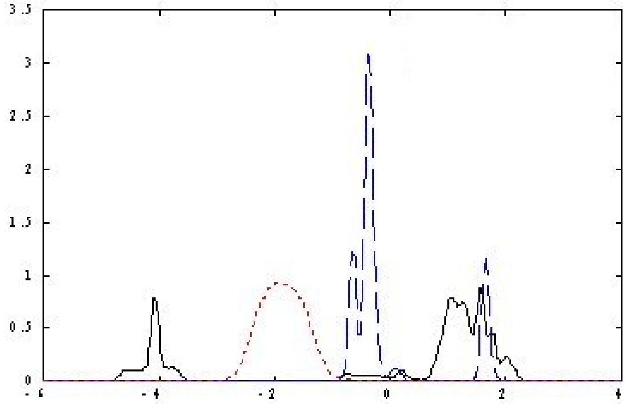


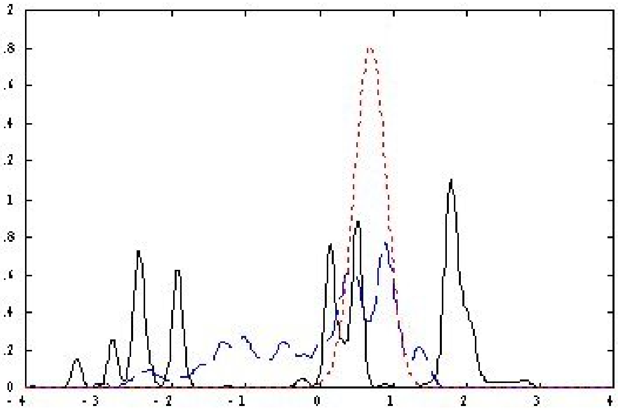




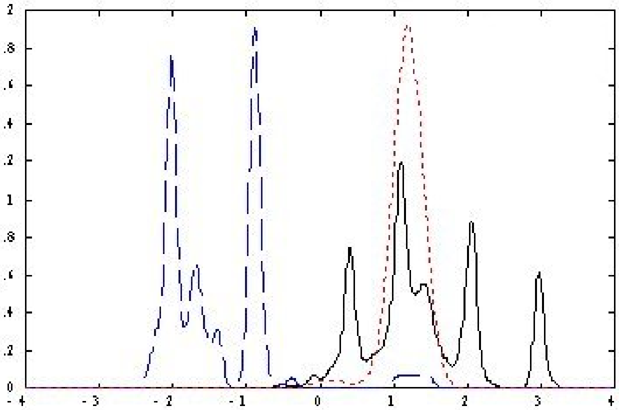
A.7 Test 3: Random Port Stream1 (Flood Category) as Unknown

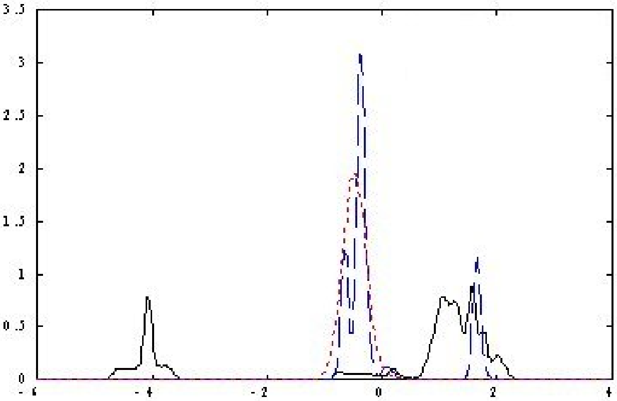


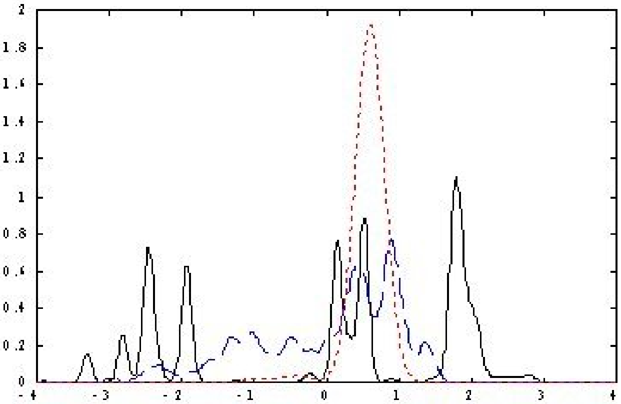




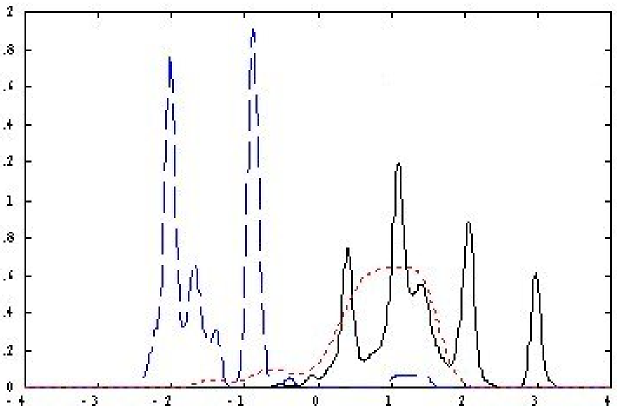
A.8 Test 3: Lower Range Nmap FIN (Scan Category) as Unknown

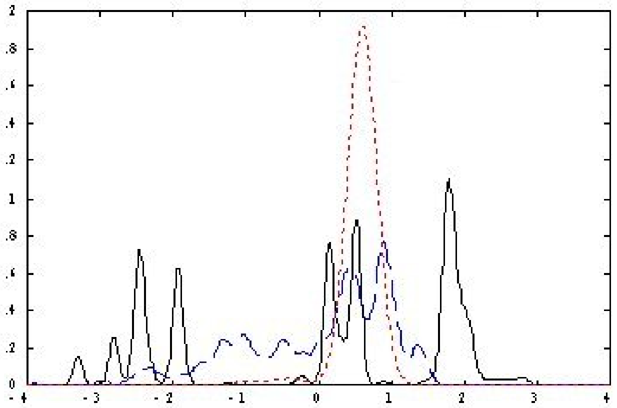


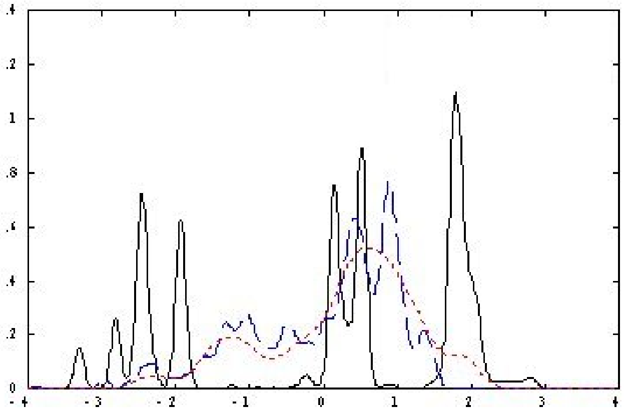




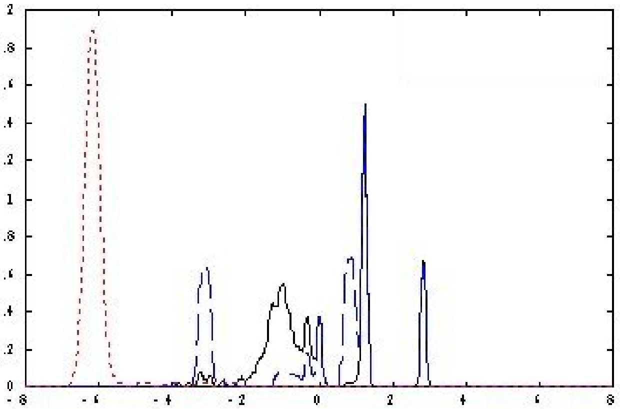
A.9 Test 3: Normal as Unknown

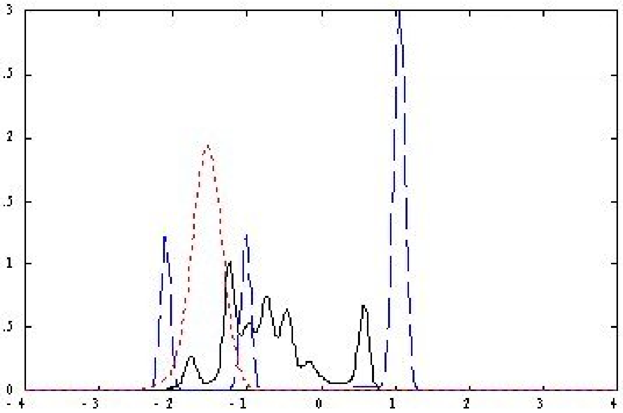


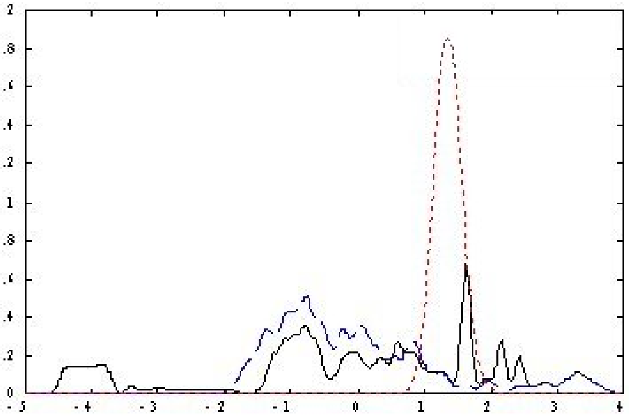




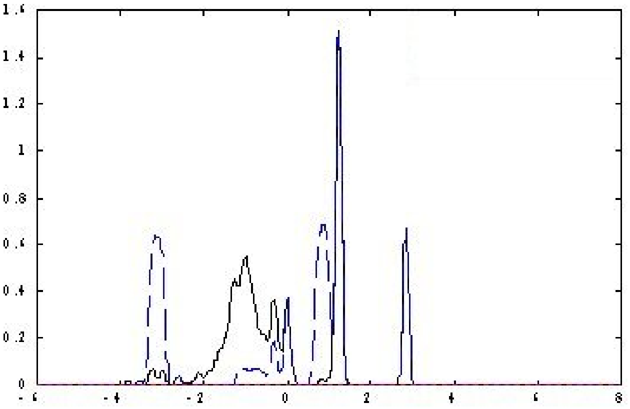
A.10 Test 4: Port 80 Synk4 (Flood Category) as Unknown

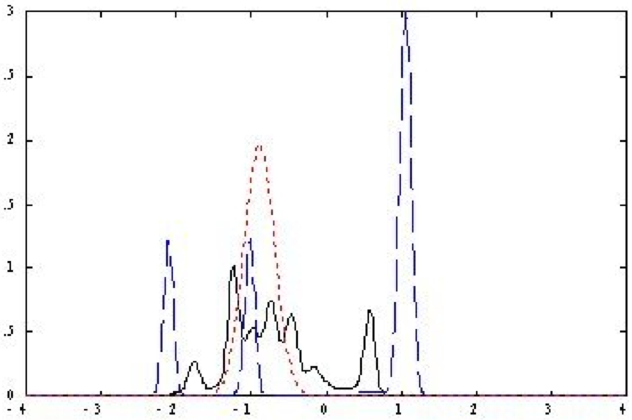


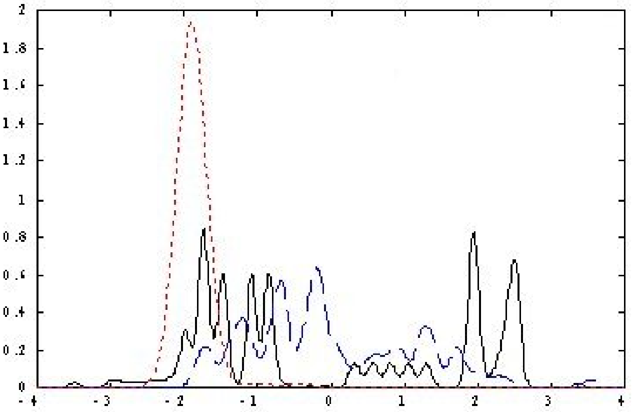




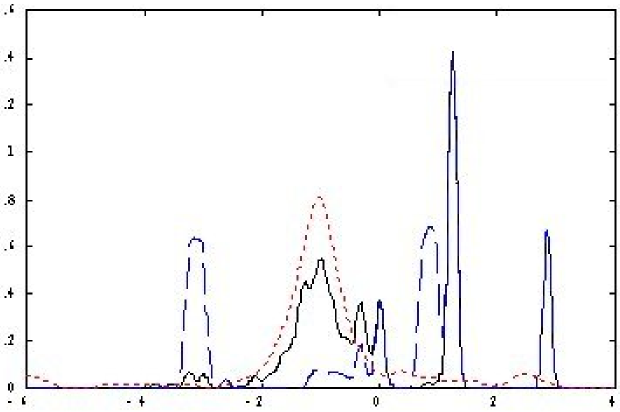
A.11 Test 4: Upper Range Superscan NULL (Scan Category) as Unknown

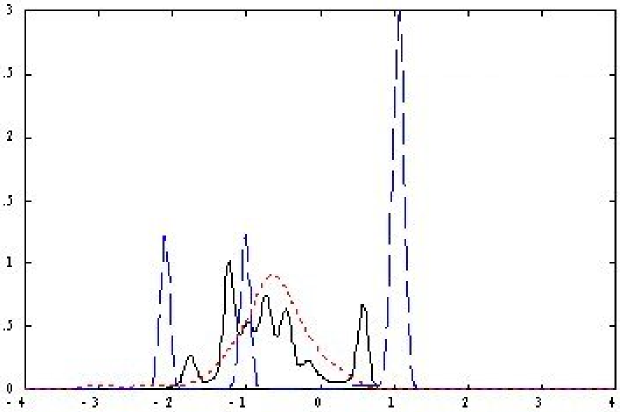


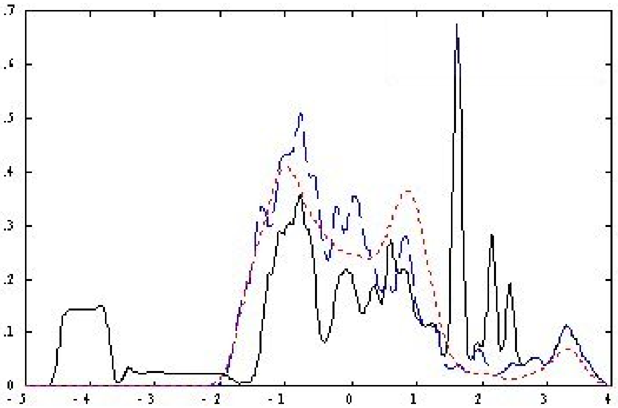




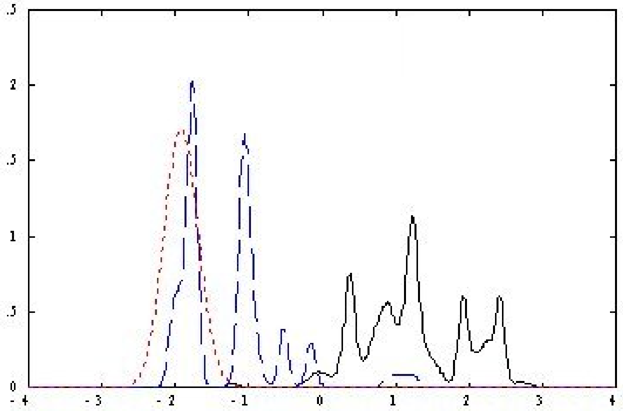
A.12 Test 4: Normal as Unknown

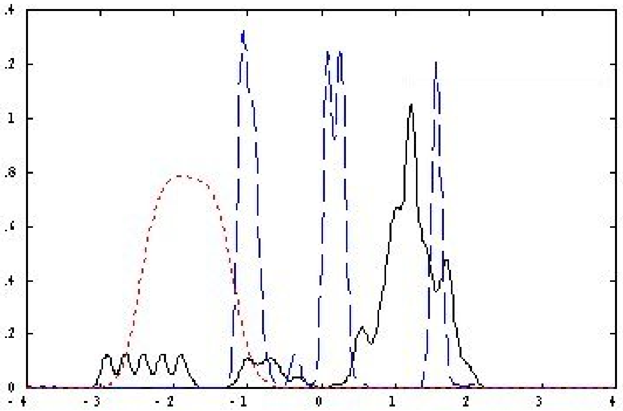


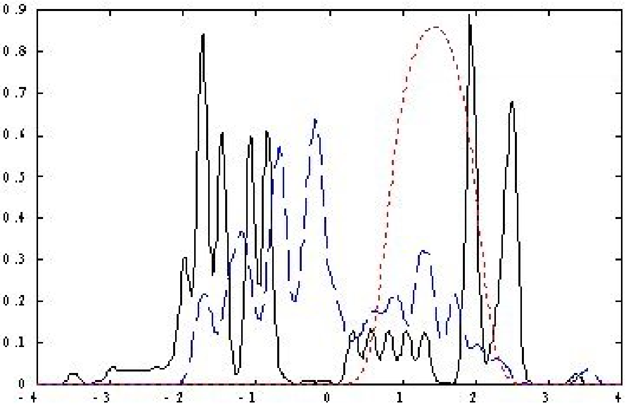




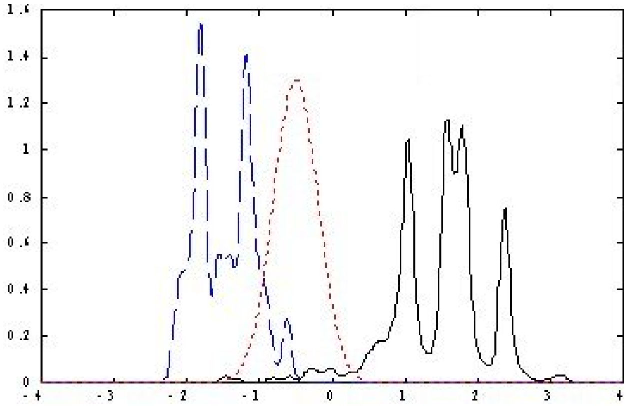
A.13 Test 5: Port 21 SynPacket (Flood Category) as Unknown

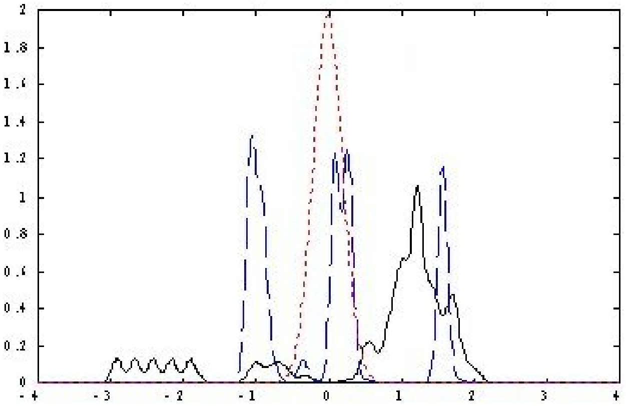


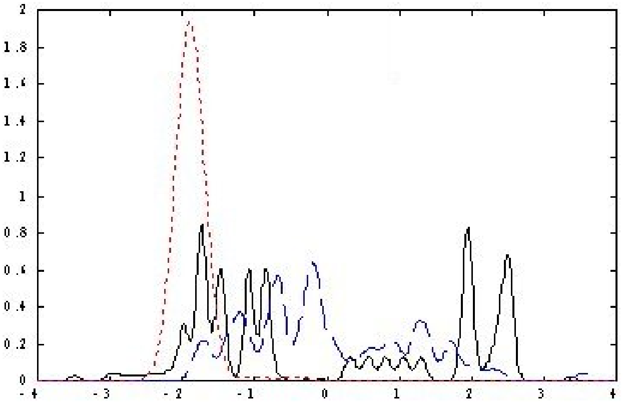




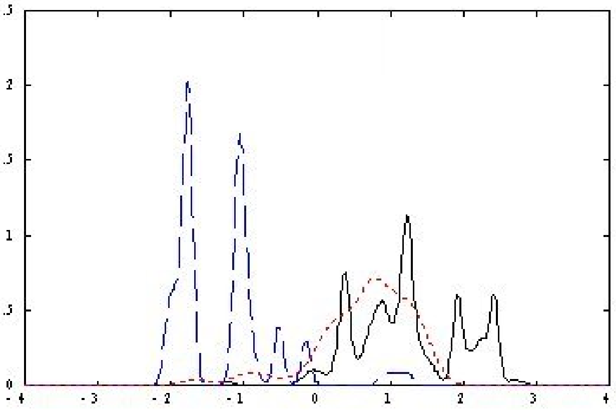
A.14 Test 5: Upper Range Nmap SYN (Scan Category) as Unknown

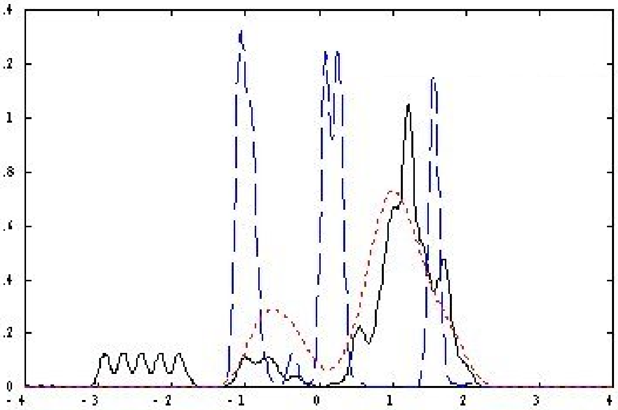


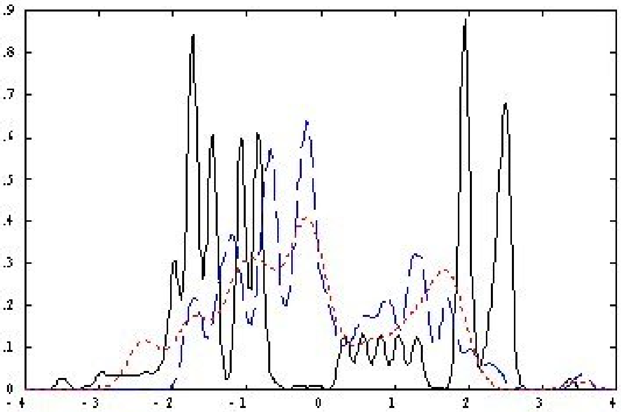




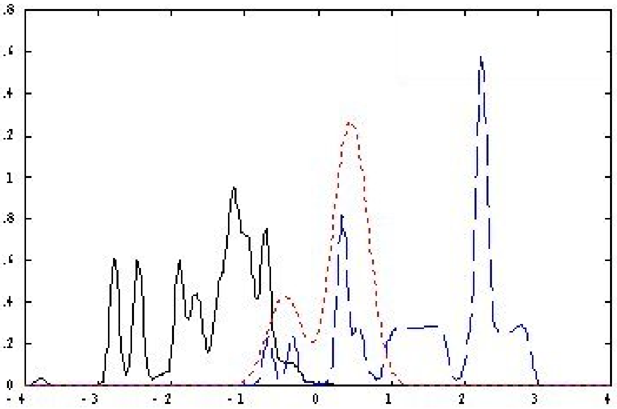
A.15 Test 5: Normal as Unknown

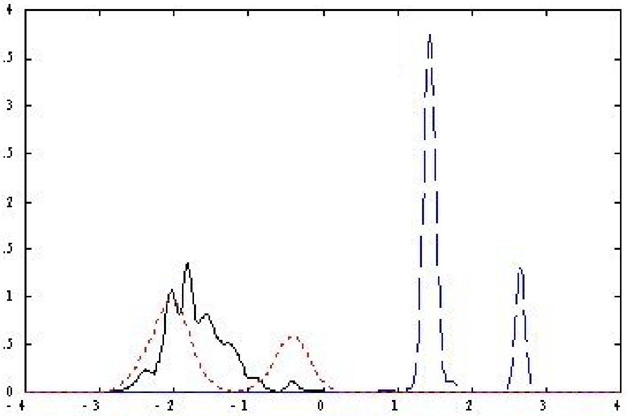


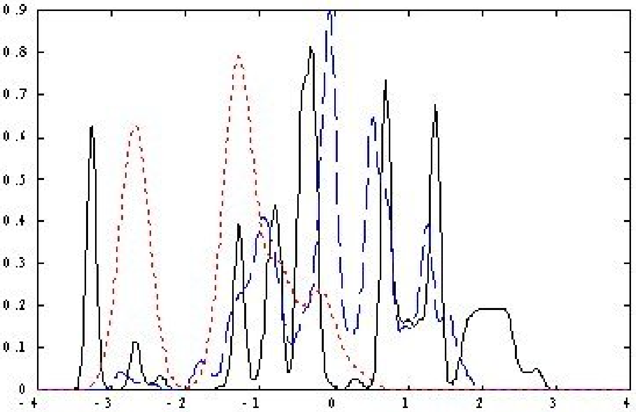




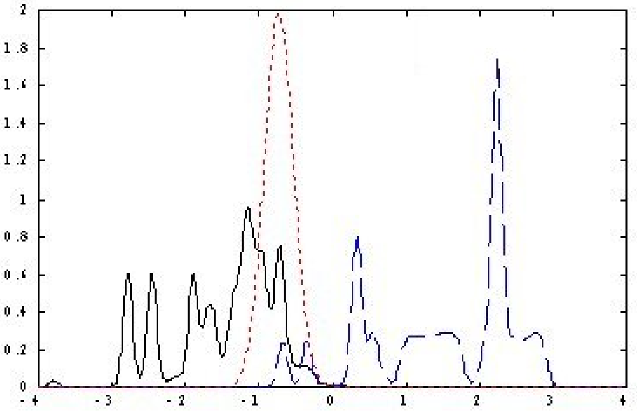
A.16 Test 6: Port 80 Hugweb (Flood Category) as Unknown

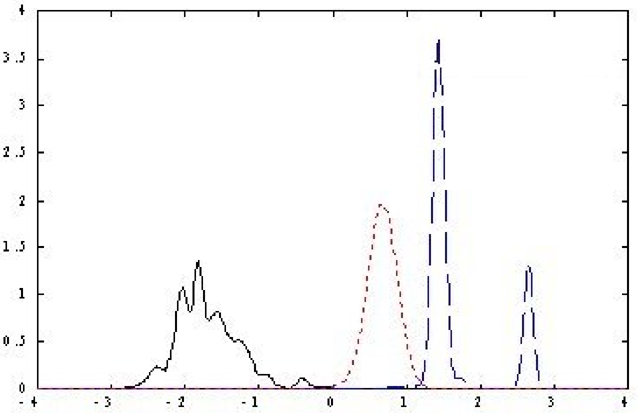


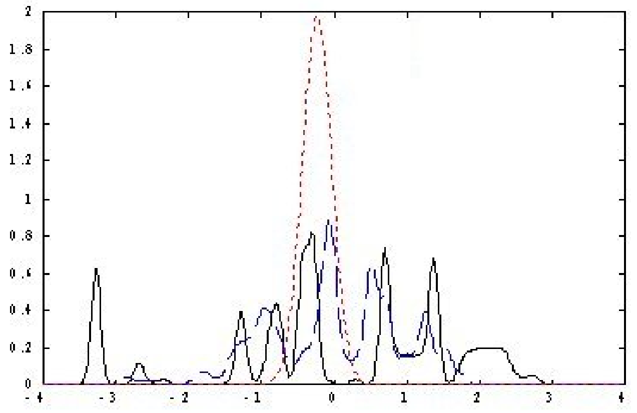




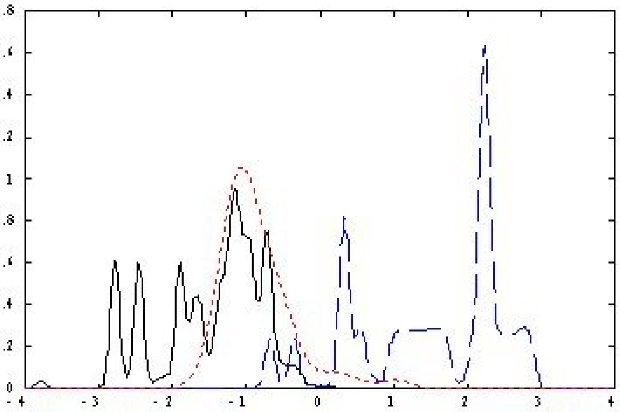
A.17 Test 6: Upper Range Elitescan (Scan Category) as Unknown

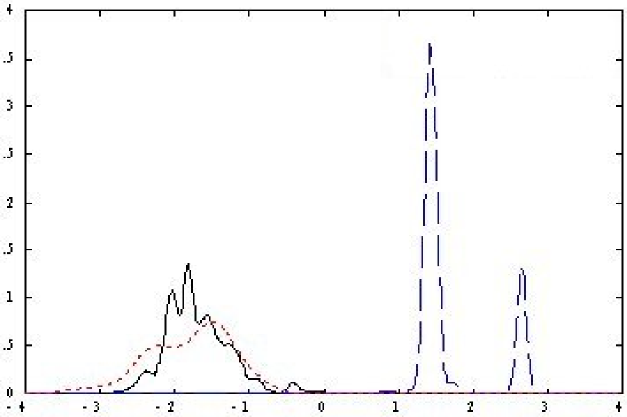


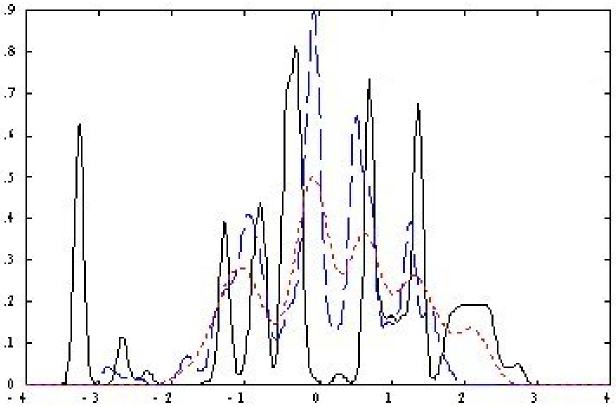




A.18 Test 6: Normal as Unknown







Appendix B. Linear Discriminant Extrapolated Results

The following results have been extrapolated from the graphs in Appendix A. If there is a clear distinction between the two categories and the majority of the unknown packets lie on one side of the discriminant vector, then the unknown packets are indicated in that category for that graph. If there is not a clear distinction, the table is marked with a ’?’. With this table, it is possible to identify the likely category. The results in these tables are estimates based on the graph results and may be open for interpretation. The first table provides a summary of the detailed information that can be found on the following table:

	Summary		
	Flood Correct	**Scan Correct**	**Normal Correct**
Test 1	Y	Y	Y
Test 2	N	N	Y
Test 3	Y	Y	N
Test 4	N	N	Y
Test 5	N	N	Y
Test 6	Y	Y	Y
Total Correct	**3**	**3**	**5**

Table B.1 Summary of linear descriminant algorithm performance based on correctly classifying unknown traffic.

Test 1 Unknown		**Test Category**	**Other Category**	**Likely Category**	**Correct**
Httpdflood	Flood	+	-	Flood	Yes
Port 80	Scan	-	+		
	Normal	?	?		
Nmap_NULL	Flood	-	+	Scan	Yes
Lower Range	Scan	+	-		
	Normal	?	?		
Normal	Flood	-	+	Normal	Yes
	Scan	-	+		
	Normal	?	?		
Test 2 Unknown		**Test Category**	**Other Category**	**Likely Category**	**Correct**
Eliteflood	Flood	-	+	Normal	No
Port 23	Scan	-	+		
	Normal	?	?		
Nmap_Connect	Flood	+	-	Unknown	No
Lower Range	Scan	+	-		
	Normal	?	?		
Normal	Flood	-	+	Normal	Yes
	Scan	-	+		
	Normal	?	?		
Test 3 Unknown		**Test Category**	**Other Category**	**Likely Category**	**Correct**
Stream1	Flood	+	-	Flood	Yes
Random Ports	Scan	? -	? +	Questionable	
	Normal	?	?		
Nmap_FIN	Flood	-	+	Scan	Yes
Lower Range	Scan	+	-		
	Normal	?	?		
Normal	Flood	-	+	Unknown	No
	Scan	?	?		
	Normal	?	?		

Test 4 Unknown		**Test Category**	**Other Category**	**Likely Category**	**Correct**
Synk4	Flood	?	?	Unknown	No
Port 80	Scan	?	?		
	Normal	?	?		
Superscan	Flood	?	?	Unknown	No
Upper Range	Scan	?	?		
	Normal	?	?		
Normal	Flood	-	+	Normal	Yes
	Scan	-	+		
	Normal	?	?		
Test 5 Unknown		**Test Category**	**Other Category**	**Likely Category**	**Correct**
SynPacket	Flood	+	-	Unknown	No
Port 21	Scan	?	?		
	Normal	?	?		
Nmap_SYN	Flood	+	-	Unknown	No
Lower Range	Scan	?	?		
	Normal	?	?		
Normal	Flood	-	+	Normal	Yes
	Scan	-	+		
	Normal	?	?		
Test 6 Unknown		**Test Category**	**Other Category**	**Likely Category**	**Correct**
Hugweb	Flood	+	-	Flood	Yes
Port 80	Scan	-	+		
	Normal	?	?		
Elitescan	Flood	-	+	Scan	Yes
Upper Range	Scan	+	-		
	Normal	?	?		
Normal	Flood	-	+	Normal	Yes
	Scan	-	+		
	Normal	?	?		

Appendix C. Radial Basis Results

The following graph lists the results of the Tuning phase tests. The first two graph lists the overall results of the tuning and testing phases. The second two graphs list the detailed results of each test in the phase.

In the first two graphs, the number tested results in the number of correct packets, the number the classifier could not determine a classification for, and the number it got wrong for that particular category. An ideal system would keep percent correct high with percent incorrect low, preferring an unknown over an incorrect packet.

Tuning Phase Totals	**Tested**	**Correct**	**Unknown**	**Percent Correct**	**Percent Unknown**	**Percent Incorrect**
Floods	132	70	40	53.03%	30.30%	16.67%
Scans	114	108	6	94.74%	5.26%	0.00%
Normal	132	125	3	94.70%	2.27%	3.03%
Trained Floods	180	178	1	98.89%	0.56%	0.56%
Trained Scans	180	180	0	100.00%	0.00%	0.00%

Testing Phase Totals	**Tested**	**Correct**	**Unknown**	**Percent Correct**	**Percent Unknown**	**Percent Incorrect**
Floods	132	44	63	33.33%	47.73%	18.94%
Scans	132	86	39	65.15%	29.55%	5.30%
Normal	132	127	1	96.21%	0.76%	3.03%
Trained Floods	180	176	0	97.78%	0.00%	2.22%
Trained Scans	180	180	0	100.00%	0.00%	0.00%

The second two graphs list the detailed results for the tuning and testing phases respectively. The tuning phase lists the sigma range where the best results were found. The testing phase utilizes the best overall sigma value from the tuning phase, or 0.7.

	Tuning Phase Results	**Left Out**	**Best Sigma**	**# Packets**	**# Correct**	**# Unknown**	**Percent Correct**
test1	Floods	hugweb_80	1.0-1.2	22	22	0	100.00%
	Scans	nmap_syn_ur	1.0-1.2	22	22	0	100.00%
	Normal	hand sampling	1.0-1.2	22	21	0	95.45%
	Trained Floods	hugweb_80	1.0-1.2	30	30	0	100.00%
	Trained Scans	nmap_syn_ur	1.0-1.2	30	30	0	100.00%
test2	Floods	eliteflood_110	0.4	22	0	18	0.00%
	Scans	ddosscan	0.4	4	0	4	0.00%
	Normal	hand sampling	0.4	22	20	2	90.91%
	Trained Floods	eliteflood_110	0.4	30	29	1	96.67%
	Trained Scans	ddosscan	0.4	30	30	0	100.00%
test3	Floods	stream1_110	1.2-1.6 best 0.7-1.1 good	22	4	0	18.18%
	Scans	nmap_xmas_lr	1.2-1.6 best 0.7-1.1 good	22	22	0	100.00%
	Normal	hand sampling	1.2-1.6 best 0.7-1.1 good	22	21	0	95.45%
	Trained Floods	stream1_110	1.2-1.6 best 0.7-1.1 good	30	30	0	100.00%
	Trained Scans	nmap_xmas_lr	1.2-1.6 best 0.7-1.1 good	30	30	0	100.00%
test4	Floods	synk4_21	0.5-0.8	22	0	22	0.00%
	Scans	elitescan_lr	0.5-0.8	22	21	1	95.45%
	Normal	hand sampling	0.5-0.8	22	21	0	95.45%
	Trained Floods	synk4_21	0.5-0.8	30	29	0	96.67%
	Trained Scans	elitescan_lr	0.5-0.8	30	30	0	100.00%
test5	Floods	synpacket_1-1024	0.7-0.8	22	22	0	100.00%
	Scans	superscan	0.7-0.8	22	22	0	100.00%
	Normal	hand sampling	0.7-0.8	22	21	0	95.45%
	Trained Floods	synpacket_1-1024	0.7-0.8	30	30	0	100.00%
	Trained Scans	superscan	0.7-0.8	30	30	0	100.00%
test6	Floods	httpdflood_80	0.7-?	22	22	0	100.00%
	Scans	nmap_null_ur	0.7-?	22	21	1	95.45%
	Normal	hand sampling	0.7-?	22	21	1	95.45%
	Trained Floods	httpdflood_80	0.7-?	30	30	0	100.00%
	Trained Scans	nmap_null_ur	0.7-?	30	30	0	100.00%

	Testing Phase	Left Out	Best Sigma	#	#	#	Percent
	Results			Packets	Correct	Unknown	Correct
test1	Floods	httpdflood_80	0.7	22	22	0	100.00%
	Scans	nmap_null_lr	0.7	22	22	0	100.00%
	Normal	hand sampling	0.7	22	20	0	90.91%
	Trained Floods	httpdflood_80	0.7	30	30	0	100.00%
	Trained Scans	nmap_null_ur	0.7	30	30	0	100.00%
test2	Floods	eliteflood_23	0.7	22	0	4	0.00%
	Scans	nmap_connect_lr	0.7	22	0	15	0.00%
	Normal	hand sampling	0.7	22	22	0	100.00%
	Trained Floods	eliteflood_23	0.7	30	28	0	93.33%
	Trained Scans	nmap_connect_lr	0.7	30	30	0	100.00%
test3	Floods	stream1_rand	0.7	22	0	22	0.00%
	Scans	nmap_fin_lr	0.7	22	20	2	90.91%
	Normal	hand sampling	0.7	22	21	1	95.45%
	Trained Floods	stream1_rand	0.7	30	30	0	100.00%
	Trained Scans	nmap_fin_lr	0.7	30	30	0	100.00%
test4	Floods	synk4_80	0.7	22	0	22	0.00%
	Scans	superscan_lr	0.7	22	22	0	100.00%
	Normal	hand sampling	0.7	22	22	0	100.00%
	Trained Floods	synk4_80	0.7	30	30	0	100.00%
	Trained Scans	superscan_lr	0.7	30	30	0	100.00%
test5	Floods	synpacket_21	0.7	22	1	14	4.55%
	Scans	nmap_syn_ur	0.7	22	0	22	0.00%
	Normal	hand sampling	0.7	22	22	0	100.00%
	Trained Floods	synpacket_21	0.7	30	28	0	93.33%
	Trained Scans	nmap_syn_ur	0.7	30	30	0	100.00%
test6	Floods	hugweb_80	0.7	22	21	1	95.45%
	Scans	elitescan_ur	0.7	22	22	0	100.00%
	Normal	hand sampling	0.7	22	20	0	90.91%
	Trained Floods	hugweb_80	0.7	30	30	0	100.00%
	Trained Scans	elitescan_ur	0.7	30	30	0	100.00%

Appendix D. Network-Based Anomaly Detection Using Discriminant Analysis

The following paper was published in December, 2001 in the Journal of Information Warfare, Volume I, Issue 2.

Network-Based Anomaly Detection Using Discriminant Analysis

George E. Noel, Steven C. Gustafson, Gregg H. Gunsch

Graduate School of Engineering and Management
Air Force Institute of Technology, Ohio
Email: george.noel@afit.edu, steven.gustafson@afit.edu, gregg.gunsch@afit.edu

Abstract

Anomaly-based Intrusion Detection Systems (IDS) can be a valuable tool for detecting novel network attacks. This paper analyzes the use of linear and non-linear discriminant analysis on packet header information from Transport and Internet layers of the TCP/IP model to classify packets as normal or abnormal. By training on normal traffic for a particular service (web and secure shell) and known attacks, the classifier can automatically identify differences between packets that may be used to classify future unknown traffic.

Keywords: Intrusion Detection, Anomaly Detection, Pattern Recognition, Discriminant Analysis

Introduction

Network attacks frequently originate from recreational hackers, but occasionally they involve seasoned cyberspace veterans working for terrorist organizations or hostile nations. Attacks utilize a myriad of vulnerabilities that are relatively easy and cost effective to exploit, yet protecting against these threats can be difficult given the hundreds of potential entry points for hackers. A layered defense may prove effective against many attacks by stopping the inexperienced or casual hacker. Firewalls provide barrier protection, while internal protection can involve closing unnecessary services. Intrusion detection systems provide a relatively new layer of defense. Although IDSs are effective against common attacks launched by inexperienced hackers, they do not easily detect novel attacks or attacks spread over long periods of time.

Most current IDSs use signature-based detection mechanisms that match incoming network packets with a signature database of known attacks. Although effective at catching attacks that have been in existence for a long time, their mechanisms fail to detect new attacks. Often it takes months before a signature is developed for a new attack and even longer before system administrators update signatures. The lack of a current signature leaves machines open to new attacks. In addition, signature databases can grow uncontrollably large which may significantly impact IDS performance (Stillerman et al., 1999).

Many approaches have been used to improve IDS detection of new and 'slow' attacks that are spread out over time. Since the first anomaly-based IDS was proposed (Denning, 1987), anomaly-detection has grown in many directions. The fundamental precept is that any network behavior that falls outside of normal boundaries is a possible attack. One approach models anomaly-detection after the human immune system, using antibodies to detect abnormal behavior much as the human immune system detects abnormal peptides (Forrest et al. 1997). The use of immune systems as anomaly-detectors is a continuing area of research (Dasgupta et al., 1997; Williams, 2001). Other approaches use neural networks to detect abnormal system activity by monitoring system calls (Ghosh et al., 1999), reviewing system audit trails (Ghosh et al., 2000), and scrutinizing network application layer data streams (Bonifácio et al., 1998).

Although there is much research on using pattern classifiers, such as neural networks, to detect abnormal system usage, very little research attempts to use pattern classifiers to detect abnormalities in network packet fields. Recent work by Lee and Heinbuch (2001) employed neural networks to detect abnormalities in network session parameters, including the number of SYN packets received, the number of FIN packets received, and the number of new connections established; however, their research does not analyze network packets for abnormalities.

This research takes a different approach to the anomaly-detection problem. It seeks to ascertain whether a pattern classification algorithm can be used to distinguish between normal traffic and network attacks for a particular Internet service. Linear discriminant analysis (with an optional nonlinear transformation) provides a simple but powerful tool for differentiating between normal and attack traffic. The linear discriminant may provide a generic detection tool for detecting new, previously unknown attacks, and may also provide information about the inputs (or features) that have the greatest impact on successful classification.

Pattern Classifier

Pattern classification involves predicting future data based on past data. It is related, in key aspects, to pattern recognition, discriminant analysis, decision theory, and assignment analysis (James 1985).

The success of a classifier depends on the data and features chosen for its implementation. Thus the selection process that precedes actual classification can be more important than selecting the 'correct' classification algorithm. Methods for classification typically involve sensing, segmentation, feature extraction, classification, post-processing, and finally decision (Duda et al., 2001).

Attacks for training the classifier must be chosen carefully. Choosing inappropriate attacks could result in a classifier that trains on insignificant features. First, the attack must consist of the same packet type as normal traffic. This research tests the system with web service and secure shell (SSH) traffic. Web and SSH traffic consist mainly of TCP packets, with the potential for a small number of ICMP packets used in reporting errors. Thus, attacks chosen should also consist mainly of TCP packets (it would be a trivial matter for a pattern recognition tool to detect UDP attacks amidst TCP normal traffic). Additionally, attacks should be directed at the same network port to prevent classifier training on the destination port number.

The classifier limitations should be considered when choosing attacks. To limit scope, this research is concerned with analyzing IP, TCP, UDP, and ICMP packet headers to detect attacks. Thus the actual packet contents are not visible to the classifier, excluding a certain class of attacks from effective classification. For instance, a buffer overflow may manifest itself in the data stream sent to a web server. Below the application layer, the buffer overflow attack may look like normal web traffic, making it impossible to properly classify the attack.

Once attacks and normal data sources are chosen, an effective method for sensing the information is needed. With other pattern recognition applications (such as face recognition), sensing can be a complicated matter, but with network traffic, sensing is far simpler. The network sensor should be designed to avoid contaminating classifier training data with traffic labeled as normal traffic when it is an attack or vice versa. For this research, unnecessary services on an isolated network are shut down to eliminate noise from routine network service communications. A Unix-based network packet sniffer called TCPDump is used to capture packet information from normal network traffic for a particular service and five attacks. The raw packet information in the TCPDump file is then presented to the classifier.

The segmentation process involves determining where each element begins and ends in the classification process. The segmentation process in face recognition, for example, involves identifying how the face is positioned—e.g. upside-down, slightly tilted, to the left or right side of the image, or shaded, and determining the boundaries of the face (even when positioned in abnormal ways). However, for network data, the packets must be clearly defined or the destination computer could not reassemble them into meaningful data, which makes the segmentation phase trivial for this particular application.

Segmentation can become a problem when examining higher layers in the network protocol. Some attacks use fragmentation to hide the attack from signature-based IDSs. The IDS must reassemble fragmented packets to discover application-layer attacks. If the destination system is configured to reject bad checksums but the IDS is not, a hacker can insert garbage information into the middle of an attack. The IDS will assemble the garbage packet into the data stream, hiding the actual attack from signature-based IDS. The end system would properly reject the bad checksum and reassemble the attack properly (Ptacek et al. 1998). Since this paper deals with the Transport and Internet layer of the TCP/IP model and fragmentation attacks fool Application-layer IDSs, fragmentation is not a significant issue.

Table 1 lists all network fields analyzed in the feature selection phase. In the feature extraction phase, the data is prepared for classification algorithm. Problems often arise when a field contains too many zeroes. This is particularly troublesome in binary flags, where a zero has greater significance than a zero in a field with a large range. Therefore, this research sets all binary fields to either 1 for true or -1 for false. There were several problems with the field ranges since some fields had a wide range while others had a very small range. For instance, in

Field Name	Range	Field Name	Range
+*IP TOS	0-255	+*TCP URG Flag	0-1
+*IP Len	0-65535	+*TCP ACK Flag	0-1
*IP ID	0-65535	+*TCP PUSH Flag	0-1
+*IP DoNotFrag Flag	0-1	+*TCP RESET Flag	0-1
+*IP MoreFrag Flag	0-1	+*TCP SYN Flag	0-1
IP FragOffset	0-8191	+*TCP FIN Flag	0-1
+*IP TTL	0-255	+*TCP Window Size	0-65535
Protocol	0-255	+*TCP Checksum	0-65535
+*IP Checksum	0-65535	TCP Urgent Ptr	0-65535
+*TCP Src Port	0-65535	UDP Src Port	0-65535
+*TCP Dest Port	0-65535	UDP Dest Port	0-65535
+*TCP Seq Num	0-4294967295	UDP Length	0-65535
+*TCP Ack Num	0-4294967295	UDP Checksum	0-65535
+*TCP Offset	0-15	ICMP Type	0-255
TCP CWR Flag	0-1	ICMP Code	0-255
+*TCP ECN Flag	0-1	ICMP Checksum	0-65535

Table 1: Network Packet Header fields analyzed in the classification tool. * indicates the field was used to classify web traffic, + indicates the field was used to classify SSH attacks. Remaining fields did not have enough variance for use in the classification.

the TCP Sequence Number, the maximum value is over 4 billion. Other fields, such as the TCP offset, have a range of only 0 to 15. This variation requires normalization of each field which involves simply subtracting the mean of a field for each data point and dividing the result by the standard deviation of the field. This normalization ensures that no fields have undo influence over others in the training process.

Some fields found in packet headers were removed initially to avoid improper training on features. Source and destination IP address were eliminated to prevent the classifier from training on an attacker's address if it differs from normal traffic. IP addresses are often useful for detecting anomalies, but they may 'overshadow' other patterns in packets. The IP Header Length and IP Version fields rarely changed from their values of 20 and 4 respectively, so they were eliminated.

Since the approach for selecting an appropriate classifier was to start with a simple design, Fisher's linear discriminant was initially chosen, and it proved to be adequate for discriminating attack data from normal data. The usual goal of a linear discriminant is to find a line through an n-dimensional space that provides an optimal discriminant between two classes.

The Fisher linear discriminant projects all points onto a single vector, essentially reducing a multidimensional problem to a single dimension. The Fisher linear discriminant can be represented by $y = w^t x$ where y is the position of the multidimensional point x projected onto the w vector, which, when divided into two segments, provides optimum classification in that the data points mapped onto it tend to be in one segment for one class and the other segment for the other class. Finding w involves maximizing the ratio of the between-scatter matrix S_B and the within-scatter matrix S_W. The between-scatter matrix measures the squared distance of the means of the classes, and the within-scatter matrix measures the sum of the variances of the classes. Thus a small within-scatter matrix and a large between-scatter matrix provides the best discrimination, and maximizing J,

$$J(w) = \frac{w^T \cdot S_B \cdot w}{w^T \cdot S_W \cdot w}$$

ensures the best linear classifier (Duda et al., 2001). The within scatter matrix, or S_W, can be calculated by summing the scatter matrix from the classes, or $S_W = S_1 + S_2$ in the case where there are two classes. The scatter matrices can be calculated with $S_w = (x_1 – m_1)(x_1\text{-}m_1)^T + (x_2 – m_2)(x_2 – m_2)^T$ where x_1 and x_2 are vectors of the multi-dimensional points in the two classes and m_1 and m_2 are the means of the classes.

Not all fields in Table 1 could be included in Fisher's linear discriminant without producing a singularity. A singularity often results when using Fisher's linear discriminant if the data is too similar. This can result in an infinite matrix when calculating S_W^{-1}. Since initial experimentation was with web traffic, there were very few ICMP packets, and most packets had all ICMP fields set to zero, which caused the within scatter matrix to be singular. To determine the fields used in the classifier, the absolute values of the normalized field elements were added together, and the fields with the largest sum and thus the greatest overall variance were used. A threshold for eliminating fields was raised until the singularity problem was resolved.

The classification phase involves using training data to find the vector that best discriminates between classes. The data points are then mapped onto the vector. Fisher's linear discriminant thus converts a 34-dimensional problem to a one-dimensional problem that can be easily visualized.

The final stage, post-processing, processes the results of the classifier. Here a non-linear transformation is performed on the resulting vector, which involves substituting the values mapped onto the discriminant vector into a polynomial equation. During development of this research, second through fifth non-linear transformations were performed on the data. The fifth degree non-linear transformation presented the best discriminant; however, the difference between the fourth and fifth non-linear transformation were small enough that a sixth degree transformation was deemed unnecessary. For the fifth degree non-linear transformation, the y =

$a + bx + cx^2 + dx^3 + ex^4 + fx^5$ maps x onto a new discriminant curve. Solving for the coefficients a, b, c, d, e, and f involves setting y to a vector that contains a 1 for normal traffic and a –1 for attack traffic. The results of Fisher's linear discriminant used for x, and the new y values are then easily calculated.

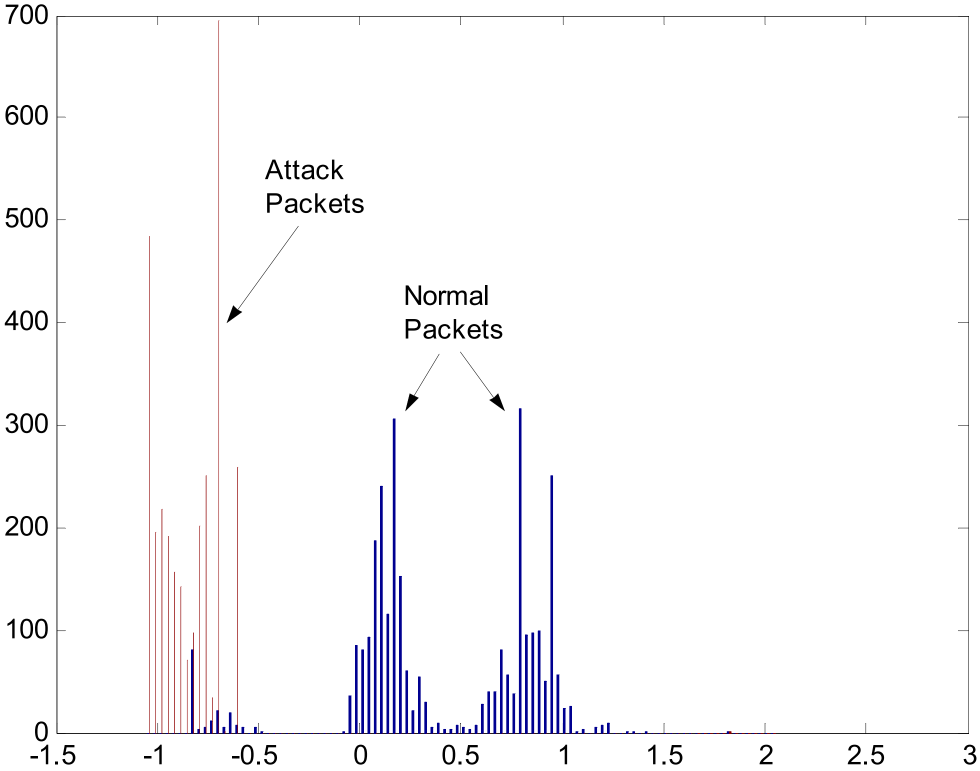


Figure 1: Histogram of normal and attack packets for web traffic after classification by Fisher's Linear Discriminant.

Results

To train the Fisher linear discriminant, 3000 normal web traffic packets were captured. In addition, packets were captured from five different attacks to total 3000 attack packets. The attacks, named httpdflood, raped, stream1, synk4, and synflood were selected for their similarity to normal web traffic and their use of Transport and Internet-layer header information to cause denial of service. They were acquired from publicly available websites and compiled on a Debian Linux system.

The 6000 points projected onto the Fisher linear discriminant vector produced the histogram in Figure 1. Here the threshold on the sum of absolute values for each field was set so that only 21 of the 32 fields were used; these fields are indicated with an asterisk in Table 1. The histogram demonstrates successful discrimination between attack packets on the left and normal packets on the right; however, a few normal packets cross into the attack domain. Though difficult to see in Figures 1 through 3, a few attack packets resemble normal packets.

Using the Gaussian technique Parzen window, the distribution of the classified packets can be easily visualized. This technique centers a Gaussian distribution on data points, multiplying the amplitude of each distribution by the number of points and sums the results. Here the window size is the constant standard deviation of each Gaussian distribution, and increasing the window size smoothes the resulting total distribution.

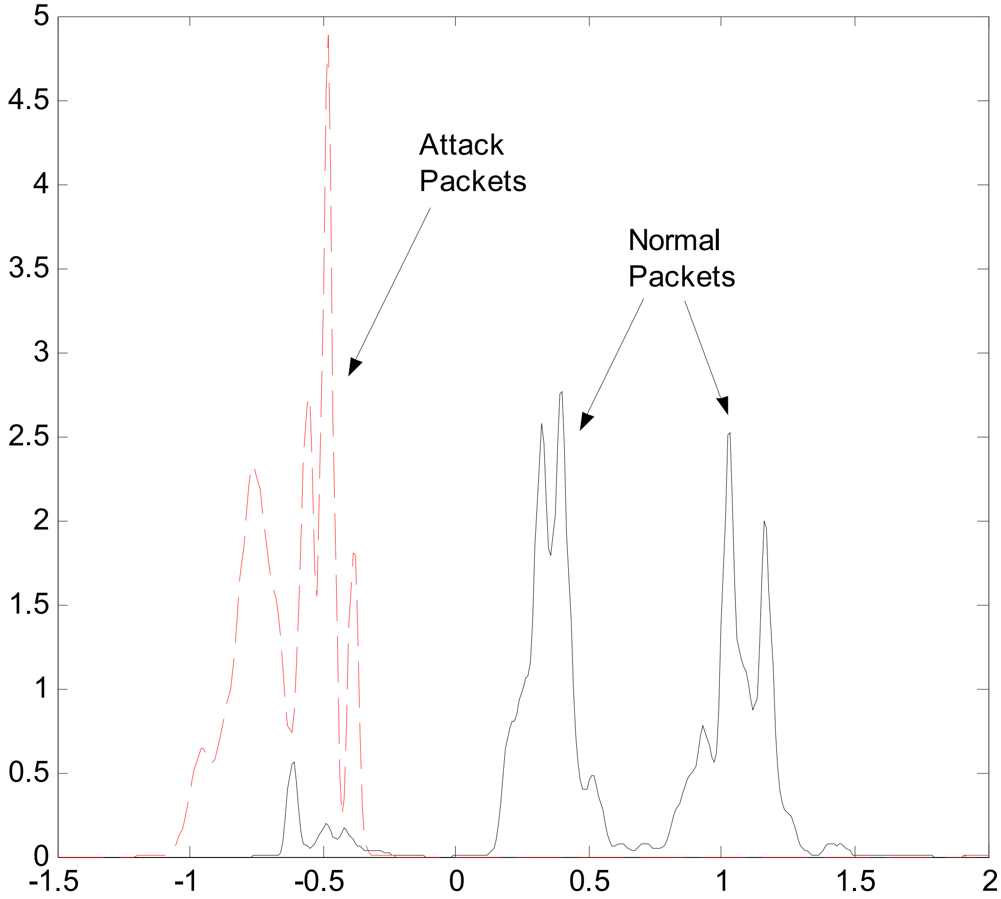


Figure 2: Gaussian Parzen Window (standard deviation 1d) probability densities of normal and attack packets for web traffic after classification by Fisher's linear discriminant

Figure 2 demonstrates the same histogram as Figure 1 but instead, uses a Parzen window with a standard deviation of 1d. Increasing the standard deviation to 4d smoothes the distribution further as shown in Figure 3.

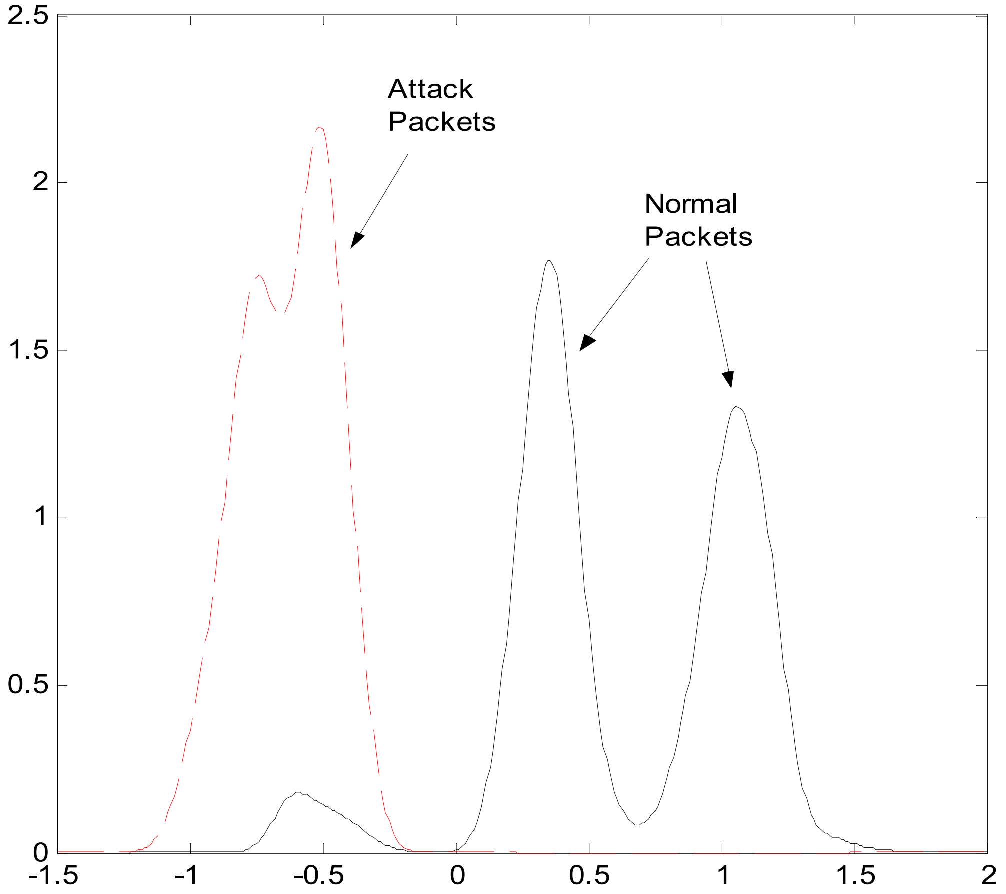


Figure 3: Gaussian Parzen Window (standard deviation 4d) of normal probability densities and attack packets for web traffic after classification by Fisher's Linear Discriminant

Figure 3 indicates that the linear classifier is effective in classifying attacks from normal web traffic, producing only a small section of false alarms. Processing the results using a non-linear polynomial transformation could potentially increase classifier accuracy. Figure 4 shows the results of such processing using a 5th degree polynomial.

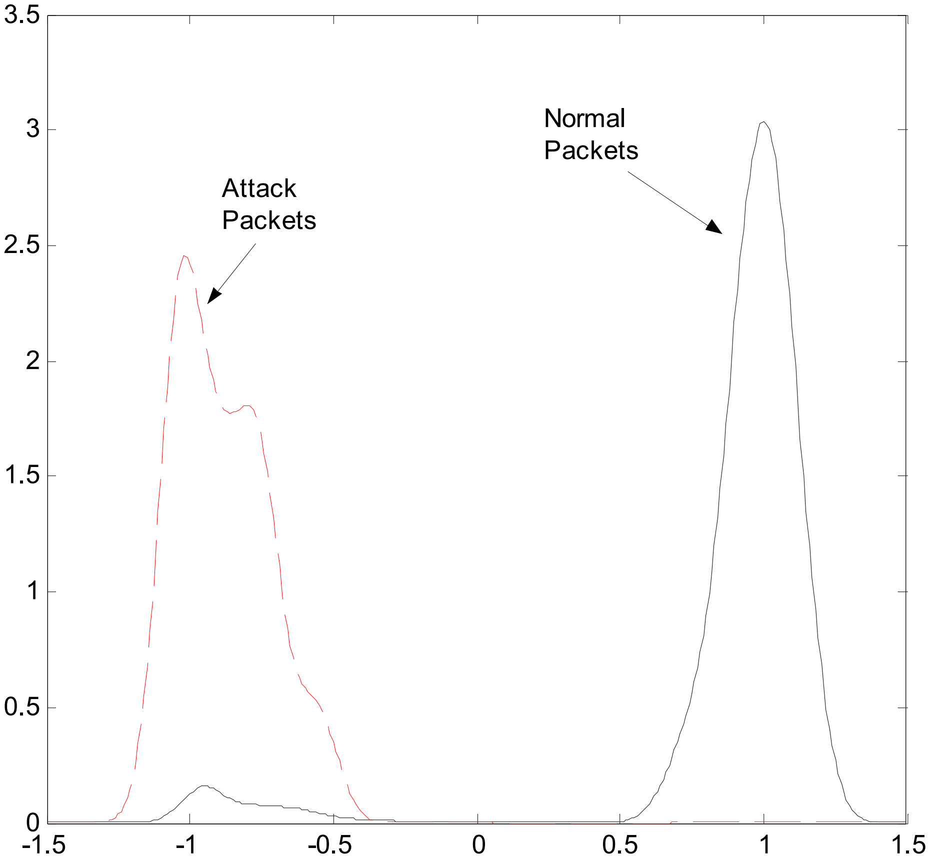


Figure 4: Gaussian Parzen Window (standard deviation 4d) of normal and attack packets for web traffic after classification by Fisher's Linear Discriminant and a Fifth-degree non-linear transformation

Figure 4 demonstrates improvement over Figure 3 in that the 'spread' between the two classes is increased, making their boundaries more distinct. A small segment of normal traffic still appears in attack traffic, threatening false alarms.

While the classification method is promising, it only demonstrates success against the chosen attacks compared to web traffic. For this approach to be useful, the classifier must detect anomalous behavior in other forms of traffic. The same five attacks were run against SSH port 22 and recorded to total 3000 attack packets. Three thousand packets were recorded from normal SSH traffic, generated by an SSH client and daemon from SSH Communications Security, and used to train Fisher's Linear Discriminant.

Initial results produced a perfect classifier that correctly classified all 6000 packets. However, this perfect classifier is unrealistic: the IP ID field was found to be the primary field that the discriminant function used to classify attacks, and the attacks had a different IP ID field than normal SSH traffic. It would be a trivial matter for an attacker to modify this field, permitting the attacks to 'slip by' the detector. For this reason, the IP ID field was removed and the packets were processed by the classifier again, which produced the results in Figure 5. This classifier uses multiple fields to discriminate between attack and normal traffic, making misclassification less likely.

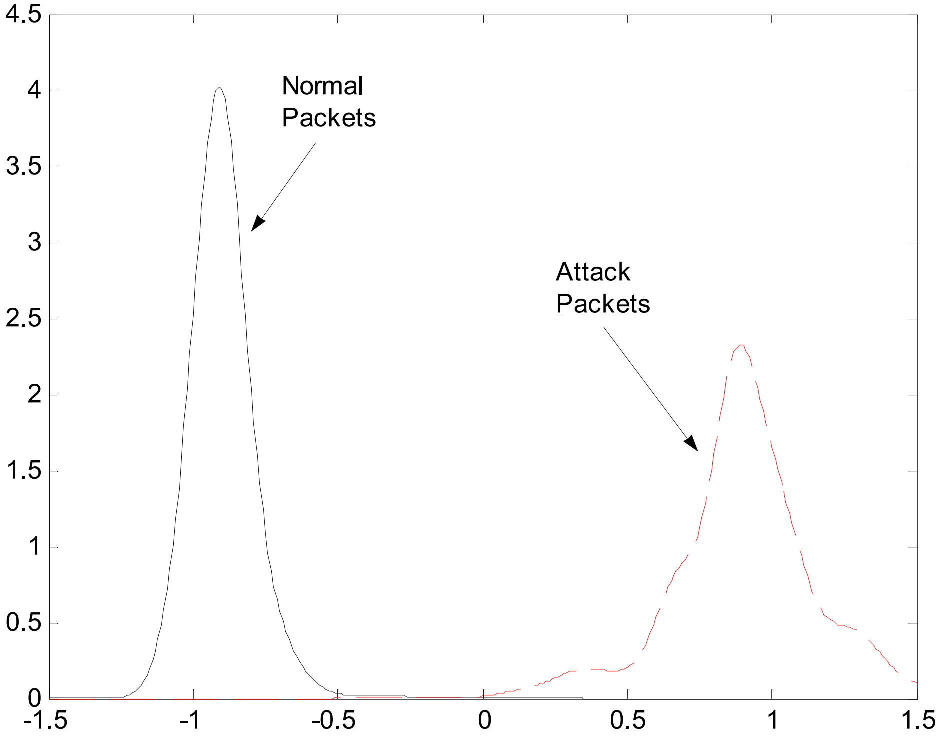


Figure 5: Gaussian Parzen Window (standard deviation 4d) of normal and attack packets for SSH traffic after classification by Fisher's linear discriminant

Before the classifier can discriminate new packets, a decision point must be established. The decision point D indicates an attack if the new packet has x < D and normal traffic if the packet has x > D, where x is the position of the projected point. The value of D may vary depending on security requirements. To avoid missing an attack at all costs, D would be further to the right, thus increasing the likelihood of detecting attacks; however, an increase in false alarms would result. In contrast, to avoid the annoyance of multiple false alarms, D would be further left.

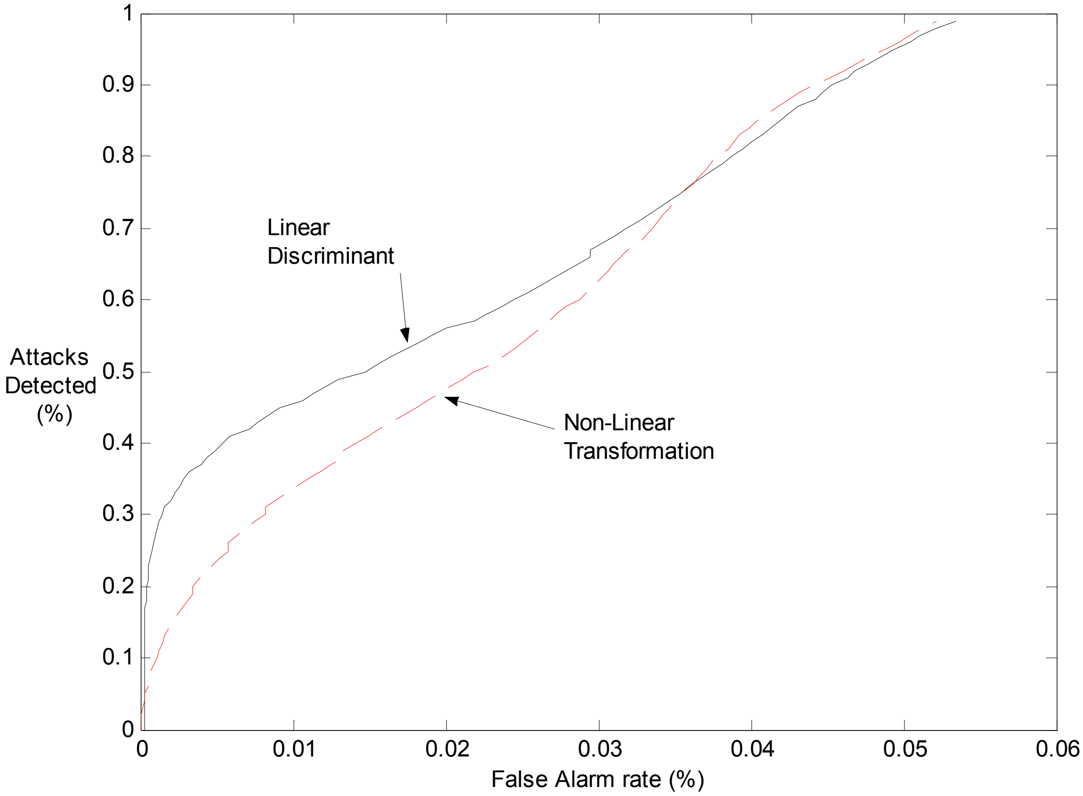


Figure 6: Receiver Operating Characteristic (ROC) curves for the Fisher linear discriminant and this discriminant with a fifth degree non-linear transformation. The ROC curves were generated using the results for web traffic shown in Figure 3 and 4.

To determine a suitable decision point, a Receiver Operating Characteristic (ROC) curve can be valuable. A ROC curve provides the percentage of attacks that can be detected for a particular false alarm rate. Figure 6 shows the web traffic ROC curve for both the Fisher linear classifier with and without the fifth-degree non-linear transformation. The non-linear transformation decreased performance given a 0% to 3.8% false alarm rate, but above 3.8%, the non-linear transformation provided a slight improvement.

Conclusion

This paper introduced simple linear discriminant analysis on network packet features to distinguish between normal and attack network traffic and demonstrated the potential for using linear discriminants to detect attacks. Some questions must be answered before a linear classifier can become a viable attack detector. First, can the same linear classifier be applied to other network services, such as telnet or e-mail? Second, would the same linear classifier be effective against other attacks that manifest themselves at or below the Transport network layer? Although care was taken to select attacks that were similar to normal web and SSH traffic, other attacks may be too similar and may “slip past” a linear discriminant-based anomaly detector. Third, would the classifier be as effective in a real-world network? The data used here was collected in a laboratory network environment to prevent unwanted network noise. A real world network would contain noise that could disrupt the training of a classifier. Finally, would the classifier be effective in detecting novel attacks? In spite of these questions, the results of the experimentation provide promising evidence that a linear discriminant can be effectively used for network anomaly detection.

References

Bonifácio, J.M., Cansian, A.M., and Carvalho, A.C.P.L.F. (1998). Neural Networks Applied in Intrusion Detection Systems. *IEEE* pp. 205-210.

Dasgupta, D., and Attoh-Okine, Nii, (1997). Immunity-Based Systems: A Survey. *IEEE,* pp. 369-374.

Denning, D.D. (1987). An Intrusion Detection Model. *IEEE Transactions on Software Engineering*, SE-13(2) pp. 222-232.

Duda, R.O., Hart, P.E., and Stork, D.G., (2001). *Pattern Classification.* John Wiley & Sons, New York.

Forrest, S., Hofmeyr, S.A., and Somayaji, A., (1997). Computer Immunology. *Communications of the ACM*, 40(10), pp. 88-96.

Ghosh, A.K., and Schwartzbard, A., (1999). A Study In Using Neural Networks For Anomaly and Misuse Detection. *Proceeding of the 8th USENIX Security Symposium,* Washington, D.C.

Ghosh, A.K., Michael, C., and Schatz, M., (2000). A Real-Time Intrusion Detection System Based on Learning Program Behavior. *RAID Conference*, pp. 93-109.

James, M., (1985). *Classification Algorithms*. John Wiley & Sons, New York.

Lee, S.C., and Heinbuch, D.V., (2001). Training a Neural-Network Based Intrusion Detector to Recognize Novel Attacks. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans,* 31(4) pp. 294-299.

Ptacek, T.H., and Newsham, T.N., (1998). Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. *Secure Networks, Inc.*

Stillerman, M., Marceau, C., Stillman, M., (1999). Intrusion Detection For Distributed Applications. *Communications of the ACM,* 7:, pp. 63-69.

Williams, P., (2001). Warthog: Towards A Computer Immune System For Detecting 'Low and Slow' Information System Attacks. MS thesis, AFIT/GCS/ENG01M-15. School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH.

Bibliography

[Bishop95] Bishop, Christopher M. *Neural Networks for Pattern Recognition.* New York: Oxford University Press, 1995.

[Bonifacio98] Bonifacio, J., et al. "Neural Networks Applied In Intrusion Detection Systems," *IEEE* (1998).

[Bow84] Bow, Sing-Tze. *Pattern Recognition: Applications To Large Data-Set Problems.* New York: Marcel Dekker, 1984.

[Duda01] Duda, Richard O., et al. *Pattern Classification.* New York: John Wiley & Sons, 2001.

[Forrest97] Forrest, Stephanie, et al. "Computer Immunology," *Communications of the ACM*, *10*:88–96 (October 1997).

[Fukunaga90] Fukunaga, Keinosuke. *Introduction to Statistical Pattern Recognition.* Boston: Academic Press, 1990.

[Ghosh99] Ghosh, A. and A. Schwartzbard. "A Study in Using Neural Networks for Anomaly and Misuse Detection." *Proceedings of the 8th USENIX Security Symposium.* August 1999.

[Hildreth01] Hildreth, Steven A. *Cyberwafare.* Technical Report Gov Docs CRS RL30735, Congressional Research Service, 2001.

[Herstein88] I.N. Herstein, David J. Winter. *Matrix Theory and Linear Algebra.* New York: Macmillan, 1988.

[James85] James, M. *Classification Algorithms.* New York: John Wiley & Sons, 1985.

[Minihan98] Minihan, K., "Hearing on Vulnerabilities of the National Information Infrastructure," June 1998.

[Noel01] Noel, George E., et al. "Network-Based Anomaly Detection Using Discriminant Analysis," *Journal of Information Warfare*, *1*:12–22 (December 2001).

[Northcutt01] Northcutt, Steven, et al. *Intrusion Signatures and Analysis.* New Riders, 2001.

[Peters99] Peters, Katerine M., "Information Insecurity," April 1999.

[Ptacek98] Ptacek, T.H. and T.N. Newsham. *Insertion, Evasion, and Denial of Service Eluding Network Intrusion Detection.* Technical Report, Secure Networks, Inc., January 1998.

[Ryan98] Ryan, Jake, et al. *Intrusion Detection With Neural Networks.* Technical Report, MIT Press, 1998.

[Lee01] S. Lee, D. Heinbuch. "Training a Neural-Network Based Intrusion Detector to Recognize Novel Attacks," *IEEE Transaction on Systems, Man, and Cybernetics – Part A: System and Humans.*, *31*:294–299 (July 2001).

[Scambray01] Scambray, Joel, et al. *Hacking Exposed: Network Security Secrets and Solutions.* McGraw-Hill, 2001.

[Staniford00] Staniford, Stuart, et al. *Practical Automated Detection of Stealthy Portscans.* Technical Report, Silicon Defense, 2000.

[Stevens94] Stevens, W. Richard. *TCP/IP Illustrated Volume 1.* Reading: Addison Wesley Longman, 1994.

[Stillerman99] Stillerman, M., et al. "Intrusion Detection For Distributed Applications," *Communications of the ACM*, *7*:63–69 (July 1999).

[Williams01] Williams, Paul D. *Warthog: Towards a Computer Immune System for Detecting "Low and Slow" Information System Attacks*. MS thesis, AFIT/GCS/ENG/01M-15, School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2001.

Vita

First Lieutenant George Noel received his commission from the United States Air Force Academy in 1998, with a Bachelor of Science in Computer Science. He served as the Chief of Network Engineering for the 319th Communications Squadron, Grand Forks Air Force Base, North Dakota from 1998 to 2000. He is pursuing his Master of Science in Information Resource Management at the Air Force Institute of Technology, Wright Patterson Air Force Base, Ohio. Upon graduation 1Lt Noel will be assigned to the Air Force Communications Agency at Scott Air Force Base, Illinois.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*
26-03-2002

2. REPORT TYPE
Master's Thesis

3. DATES COVERED *(From - To)*
Jun 2001 - Mar 2002

4. TITLE AND SUBTITLE
CLASSIFYING NETWORK ATTACKS USING PATTERN CLASSIFICATION ALGORITHMS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
George E. Noel III, 1st Lt, USAF

5d. PROJECT NUMBER
01-179

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P. Street, Building 640
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GIR/ENG/02M-03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/IFGB
Attn: Mr. John Feldman
525 Brooks Rd.
Rome, NY 13441-4505 Commercial: (315) 330-2664

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Information systems are often inundated with thousands of attack alerts and are unable to distinguish novice hacker probes from genuine threats. Pattern classification can help filter relatively benign attacks from alerts generated by anomaly detectors, limiting the number of alerts requiring attention.

This research investigates the feasibility of using pattern classification algorithms on network packet header information to classify network attacks. Both linear discriminant and radial basis function algorithms are trained using flood and scan attacks. The classifiers are then tested with unknown floods and scans to determine how well they categorize previously unseen attacks. The results indicate the radial basis function algorithm classifies most packets correctly. The linear discriminant algorithm correctly classifies half of unknown packets.

15. SUBJECT TERMS
intrusion detection, pattern classification, computer security, anomaly detection, linear discriminant, radial basis function, machine learning

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Gregg H. Gunsch, Ph.D., ENG
U	U	U	UU	115	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 4281